IEEE
INTERNET OF THINGS JOURNAL

IEEE Xplore® Digital Library
http://ieeexplore.ieee.org/Xplore

FlowGuard: An Intelligent Edge Defense Mechanism Against IoT DDoS Attacks

Yizhen Jia, Fangtian Zhong, Arwa Alrawais, Bei Gong, and Xiuzhen Cheng *Fellow, IEEE*

***Abstract*—IoT devices are getting more and more popular in recent years and IoT networks play an important role in industry as well as people's activities. On one hand, they bring convenience to every aspect of our daily life; on the other hand, they are vulnerable to various attacks that in turn cancels out their benefits to certain degree. In this paper, we target the defense techniques against IoT DDoS attacks and propose an edge-centric IoT defense scheme termed FlowGuard for the detection, identification, classification, and mitigation of IoT DDoS attacks. We present a new DDoS attack detection algorithm based on traffic variations and design two machine learning models for DDoS identification and classification. To demonstrate the effectiveness of the two machine learning models, we generate a large dataset by DDoS simulators BoNeSi and SlowHTTPTest, and combine it with the CICDDoS2019 dataset, to test the identification and classification accuracy as well as the model efficiency. Our results indicate that the identification accuracy of the proposed LSTM is as high as 98.9%, which significantly outperforms the other four well-known learning models mentioned in the most related work. And the classification accuracy of the proposed CNN is up to 99.9%. Besides, our models satisfactorily meet the delay requirements of IoT when deployed in edge servers with computational powers higher than a personal computer.**

***Index Terms*—IoT Security; DDoS Attacks; Edge Computing; Artificial Intelligence.**

I. Introduction

Internet of Things (IoT) is an emerging technology that offers a unified connection of diverse devices to provide services and automate the operations in different domains ranging from our daily life to crucial infrastructure systems. According to a recent statistical study [1], the market size of global IoT should reach 292 billion US dollars by 2020 and is projected to reach 1,567 billion US dollars by 2025.

Even though the market sharing and service domains of IoT are growing rapidly, their security conditions remain unpleasant. Due to the limited computational power and constrained storage of IoT devices, it is extremely hard to implement practical protection mechanisms on them. As a result, simplified lightweight protocols are generally adopted in IoT systems, making them susceptible to various attacks as inadequate level of protections can be easily bypassed by attackers. More seriously, attackers can exploit the vulnerabilities of IoT and create botnets to launch severe DDoS (Distributed Denial of Service) attacks without much effort. A DDoS attack is one in which an attacker attempts to obstruct the legitimate operations of a device and makes the device unavailable to legitimate users via extreme resource consumptions by distributed attack sources. For example, the Mirai virus infected 65,000 devices within the first 20 hours of its release in August 2016, forming a botnet to attack more IoT devices [2]. In November 2016, the smart heating systems of two buildings in Finland were shut down due to a DDoS attack caused by a Mirai botnet, which made the heating controllers to continually reboot until the heating systems came into crashes. Similar DDoS events [3] occurred worldwide after more Mirai variations were developed. Such examples indicate that IoT devices can be employed as DDoS weapons to collaboratively create botnets after being compromised by attackers; they can also become DDoS victims when their normal operations are interrupted by IoT DDoS botnets.

DDoS attacks targeting IoT devices can cause severe corruptions to the IoT meta-systems. According to a recent report by Security Today and Threat Post, the number of connected IoT devices would be around 31 billion by the year 2020, and over half of them are insecure and vulnerable to multiple attacks [4] [5]. Vulnerable IoT devices are the potential attack targets to form botnets, which then threaten the IoT system security via DDoS attacks. Moreover, the bigger the botnet, the more powerful the attack can be. Even with only a small fraction of the IoT devices are compromised to generate DDoS packets, the magnitude of the launched attacks would be enormous and such attacks could easily disrupt any target including powerful servers equipped with heavily-protected strategies.

DDoS attacks have been extensively studied in traditional network environments and numerous effective defensive methods based on powerful resources have been developed. Nevertheless, these protective strategies are largely insufficient and impractical for IoT; therefore efficient and powerful DDoS defensive schemes are desperately needed to prevent IoT from DDoS attacks. In this paper, we propose FlowGuard, an effective DDoS attack detection, identification, classification, and mitigation scheme leveraging two cooperated machine learning techniques, namely LSTM and CNN. FlowGuard consists of two major components: Flow Filter and Flow Handler, with the former maintaining the flow filtration rules generated by the Flow Handler and taking charge of DDoS attack detection while the latter analyzing suspicious flows for DDoS attack identification and classification, and filtration rule generation making use of self-evolving machine learning

Yizhen Jia, Fangtian Zhong (corresponding author), and Xiuzhen Cheng are with the Department of Computer Science, The George Washington University, Washington DC, USA.
E-mails: {chen2015,squareky_zhong,cheng}@gwu.edu
Arwa Alrawais is with the College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia.
E-mail: a.alrawais@psau.edu.sa
Bei Gong is with the school of Information Technology, Beijing University of Technology, Beijing 100124, P.R. China.
E-mail: gongbei@bjut.edu.cn

Digital Object Identifier: 10.1109/JIOT.2020.2993782

2327-4662 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

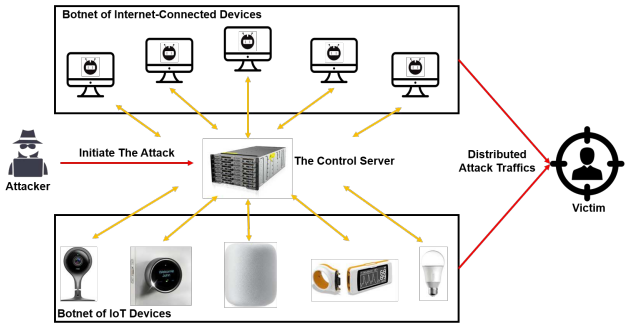


Fig. 1: Overview of DDoS Attacks

algorithms. FlowGuard is operated at the edge servers closest to the IoT network, and all the packets passing through the edge servers should be inspected by the Flow Filter, which passes the flows that are not deemed malicious based on the filtration rules. We evaluate FlowGuard with a mixed attack dataset that contains various types of DDoS attacks, and our results indicate that FlowGuard could achieve accuracies over 98.9% in attack identification and over 99.9% in attack classification without obviously affecting regular network operations.

The rest of the paper is organized as follows. Section II introduces the mainstream DDoS attacks and the state-of-the-art defense mechanisms. Section III presents our network model and threat model. Section IV proposes FlowGuard and details its design. Section V evaluates the performance of FlowGuard and Section VI concludes the paper.

II. Background and Related Work

A. DDoS Attacks

DDoS attacks refer to the family of attacks in which an attacker attempts to obstruct or stop legitimate users from accessing specific network services or resources by distributed attack sources. As shown in Fig. 1, an attacker can create a botnet by compromising vulnerable Internet-connected devices as well as IoT devices, and launches attacks by controlling the botnets through a control server; as a consequence, the victim receives massive source-varied attack traffics from the distributed compromised devices, disrupting its normal activities.

Compared to conventional DDoS attacks, IoT DDoS attacks are even harder to be defended against since IoT devices have limited storage and computational resources. An attacker can easily forge malicious packets based on the design flaws of IoT device firmware or the bugs in the communication protocols. On the other hand, besides being the victims of DDoS attacks, IoT devices can also be used as a powerful DDoS attack assistant tool. The dumb security implementations of IoT devices make them easy to be controlled by an attacker to establish a distributed botnet, which is deemed as a dominant mechanism in establishing flooding-based DDoS attacks. For example, attackers built the infamous Mirai Botnet by first taking control of 65000 IoT devices within 20 hours after its initial release, and then exploiting these compromised IoT devices to launch DDoS attacks targeting IoT networks such as OVH and Dyn [2]. After a careful exploration of existing

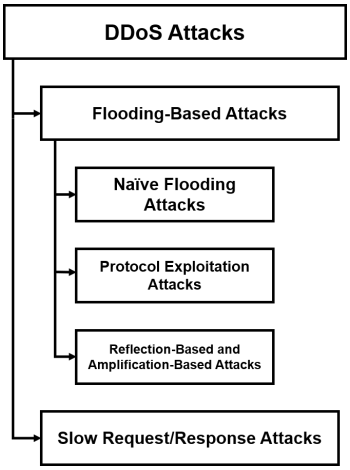


Fig. 2: DDoS Attacks Taxonomy

IoT DDoS attack scenarios, we taxonomy the corresponding DDoS attacks into two major categories, namely flooding-based attacks and slow request/response attacks, as shown in Fig. 2. Note that many researchers have proposed various classifications of common DDoS attacks [6] [7] [8] [9] [10] but we focus on IoT DDoS attacks in this paper.

Flooding-based Attacks. This type of attacks aims to exhaust a victim's resources and/or the network bandwidth based on massive disguised network packets. According to the distinction of the exploited vulnerabilities, it can be further divided into three subcategories:

- **Naïve Flooding Attacks.** The malicious packets employed for naïve flooding attacks are always composed of a large number of repetitive communication requests, which eventually fill the victim's buffer. Once the buffer is full, the victim can not accept new messages and the requests from legitimate users are then disrupted. For example, in a UDP flooding attack, an attacker can send a large amount of IP-spoofed UDP packets to random ports on the victim host. Then the victim replies with an ICMP Destination Unreachable packet if no application is listening on that port. As a result, the victim is forced to persistently send ICMP packets, eventually leading to services unreachable by the legitimate users. Naïve Flooding attacks include UDP flooding attacks, ICMP flooding attacks, DNS flooding attacks, just to name a few.
- **Protocol Exploitation Attacks.** This type of attacks is triggered by protocol features and implementation bugs. Attackers can exploit specific features and run out of resources of the victim's network. For example, the feature of the three-way handshake in TCP can be leveraged to disrupt legitimate connections by TCP SYN flooding, ACK & PUSH ACK flooding, etc. Besides, the inherent weakness of the three-way handshake connection setup process makes TCP SYN flooding attacks very possible. An attacker can initialize the attack by sending a large number of SYN messages to a server, but refusing to react the responses from the server, making the server persistently waiting for the non-existent ACK messages.

Finally, the limited buffer queue of the server is exhausted, thus no more new connection requests can be processed.

- **Reflection-based and Amplification-based Attacks.** In Reflection-based flooding attacks, attackers send forged request packets with a manipulated source address to a server. As the server is unable to distinguish the spoofed packets from legitimate ones, massive response packets are then directed to the victim specified by the manipulated source address from the server. Amplification-based flooding attack is a type of reflection-based attacks in which attackers aim to fraud the server to generate a high volume of response packets to a victim with limited requests. Attackers usually combine those two types of attacks. For example, an attacker can send a source address spoofed request packet to a DNS server by exploiting the cryptographic feature of the DNS security extension to increase the packet size. Through this method, a DNS request message of 60 bytes can be amplified to a response message over 4000 bytes in length to the victim.

Slow Request/Response Attacks. In this type of attacks, attackers hold the communication channel and exhaust the resources of a victim by spoofing high-workload requests or responses. For instance, an attacker can establish a valid HTTP connection with a victim device. Then the attacker segments the legitimate HTTP packets into tiny fragments and sends those fragments as slow as possible within the maximum allowed communication time. By setting up multiple such sessions to the victim, all vacant sockets of the victim can be finally taken up by those forged HTTP requests and as a consequence the victim becomes unavailable.

B. DDoS Defense Techniques

In order to make servers steadily serve their legitimate users, researchers came up with many techniques such as IP traceback, packet marking, entropy variations, intrusion detection and prevention, etc., to prevent DDoS attacks before they occur.

IP Traceback Techniques. For a DDoS attack, higher detection accuracy can be achieved at the closest vicinity of the victim due to the presence of aggregated attack flows, in comparison with remote disjoint attack flows. Similarly, it is desirable to perform filtering of the attack traffics closer to their attackers to avoid influence on other network users. For the purpose of packet filtration, revealing the attack source and path followed by attack packets is essential. Many researchers adopted IP traceback techniques to assist in packet filtration [11]. For example, Burch in 2000 developed a link testing tecnique that performs a recursive search from a victim until the attacker is reached [12]. The path of the link testing starts from the router closest to the victim and ends until the source router of the attacker is identified. After continuously iterating on the determination of incoming links, the node in the path at each upstream level can be revealed. Snoeren *et al.* in 2001 proposed a hash-based IP traceback approach by using a practical packet logging [13], which is aimed at storing packet digests on intermediate routers. Each router stores the digests of the packets passing through it and a digest could be the hashed value of the IP header fields of a packet. The network path is then determined by using the stored information. This technique allows us to eliminate all but a handful of physical networks that could be the source of the attacking packet stream, which can be used to defend against reflection-based and botnet-based DDoS attacks [14].

Entropy Variations. Nychis *et al.* analyzed the detection capabilities and correlations of different entropy-based metrics such as flow-header features and behavioural features [15]. Flow-header features involve IP addresses, ports, and flow-sizes, and the behavioral features are related to the number of packets during communications between nodes. By finding the difference in the entropy values of attack packets at the source or destination IP address and the corresponding port, slow request or response attacks can be detected [16]. In [17], the entropy is classified into long term entropy when the number of packets is more than 10,000 and short term entropy when it is less than 10,000. The short term entropy is used for early detection while the long term entropy is employed to classify attacks. Such an approach can be employed to identify protocol exploitation flooding attacks and amplification-based flooding attacks with appropriate thresholds.

Intrusion Detection and Prevention Systems. Roschke *et al.* in 2009 proposed a deployment architecture of Intrusion Detection Systems by taking a cloud computing paradigm, which includes a system layer, a platform layer, and an application layer. IDS is deployed on each layer of the cloud to gather alerts from sensors and the central management unit in the cloud correlates and analyzes the alerts [18]. Rengaraju *et al.* presented a distributed firewall with an Intrusion Prevention System (IPS). The firewall is deployed on the switches in SDN. These switches monitor the passing traffics for anomaly detection, and suspicious packets are transferred to the firewall for further analysis. Once a malicious activity is identified, the IPS can prevent intrusions by sending alarms, dropping packets, resetting connections, or blocking the traffics from the attackers. Then, the SDN controller updates the firewall rules. This IPS system was investigated for naïve flooding attacks [19].

The defense techniques mentioned above play their roles at different layers of the network stack. Specifically, IP traceback techniques and entropy variations are used at the network layer, while intrusion detection and prevention systems are deployed on the cloud and switches.

C. Machine Learning Based DDoS Defense Techniques

A variety of machine learning algorithms have been developed in defense of DDoS attacks. In [20], the authors proposed an effective detection scheme based on KNN with correlation analysis to detect DDoS attacks. In [21], a detection mechanism based on a One-Class Support Vector Machine (OC-SVM) was introduced for application-layer DDoS attacks, specifically for HTTP flooding attacks, SYN flooding attacks, and NTP amplification attacks. As Vishwakarma and Jain suggested in [22], honeypots with machine learning-based

approaches are effective in detecting botnet-based DDoS attacks in IoT. They used heterogenous IoT honeypots to capture device malware installation attempts and adopted unsupervised machine learning techniques such as clustering and anomaly detection to automate the process of detection and prediction of the incoming security threats by extracting features from honeypots. Asad *et al.* proposed a detection mechanism by relying on artificial neural networks with feed-forward and back-propagation algorithms to accurately discover several application-layer DDoS attacks [23].

A deep Bidirectional Long Short Term Memory-based Recurrent Neural Network model was developed in [24] to detect botnet activities within consumer IoT networks by focusing on text recognition at the packet level. Besides, Meidan, Bohadana, and Mathov adopted deep autoencoders called N-BaIoT to detect anomalous network traffics from compromised IoT devices, and nine commercial IoT devices were infected with two widely known DDoS attacks, namely Mirai and BASHLITE, to prove the model's effectiveness.

Doshi and Feamster used K-nearest neighbors, support vector machines, decision trees, random forests, and neural networks to perform data collection, feature extraction, and binary classification of network traffics [25]. These models are operating on network middleboxes (e.g. routers, firewalls, or network switches) and the corresponding devices that may be part of an ongoing DDoS attack [25]. Additionally, in [26], a DDoS attack detection and warning system was designed based on a multi-channel convolutional neural network by splitting up features in light of dimensions of time, space, packet, etc.

The machine learning algorithms mentioned above build mathematical models based on sample data, known as "attack packets", to make predictions to potential DDoS attacks. To achieve effective detection and explore their feasibility, these models were deployed on different devices such as honeypots, routers, network switches, and firewalls. In this paper, we present FlowGuard, an intelligent edge defense mechanism against IoT DDoS attacks. Compared to the above-mentioned classical and machine learning-based techniques, FlowGuard does not involve any complicated system nor does it put computational pressure on IoT devices. Nevertheless, it achieves high DDoS attack detection and classification accuracy, thus is particularly suitable to defend against IoT DDoS attacks.

III. Network Model and Threat Models

In this section, we present our network model as well as the threat model under our consideration.

A. Edge-IoT Architecture

The concept of edge computing was initially proposed by Microsoft in 2008 to reduce the latency and bandwidth by moving microdata centers closer to the information generation sources. However, this definition is not suitable for the IoT environments. In 2013, the Pacific Northwest National Laboratory (PNNL) presented a mature concept of edge computing based on the definition of Microsoft, which states that an edge computing architecture consists of two components, i.e.,

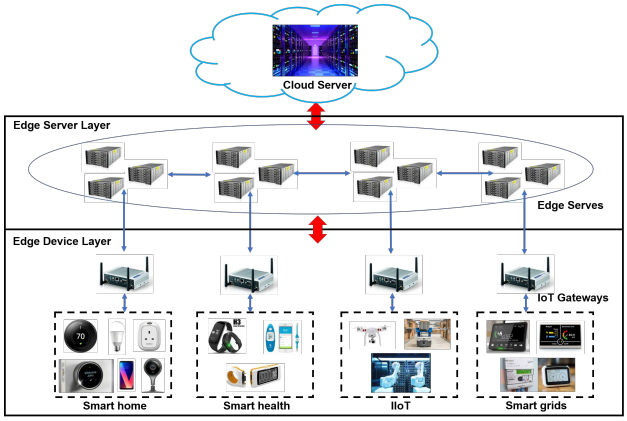


Fig. 3: The Network Model of Edge-IoT Architecture

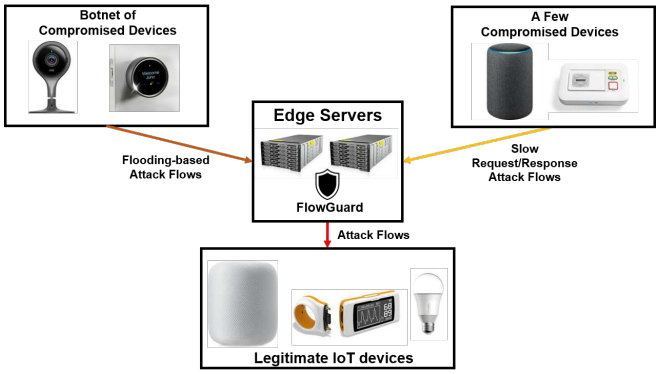


Fig. 4: The Threat Model of Our Architecture

edge servers and edge devices, with edge devices offloading their data and computational tasks to the edge servers while the edge servers processing those computational tasks based on the collected data and returning the results back to the edge devices in a real-time manner [27]. This state-of-the-art edge computing architecture meets the requirements of IoT, as shown in Fig. 3.

In this network model, edge devices refer to the IoT end devices which take the responsibility of sensing, actuating, and controlling. Edge servers are core processing and computing units with powerful computing capability and abundant memory resources located at the edge of the network. Edge servers collect the data flowing from the IoT devices and send the processed results back to IoT devices through various network interfaces. All of the computation and analysis tasks are offloaded to the edge servers from the IoT devices. IoT devices take actions based on the received computational results from the edge servers. Note that edge servers could be organized hierarchically with those closer to the IoT devices having less computational and storage resources while those closer to the cloud servers possessing more computational and storage resources.

B. IoT DDoS Threat Models

As mentioned in Section II, major IoT DDoS attacks either are flooding-based or belong to the Slow Request/Response attack category. Flooding based DDoS attacks are usually

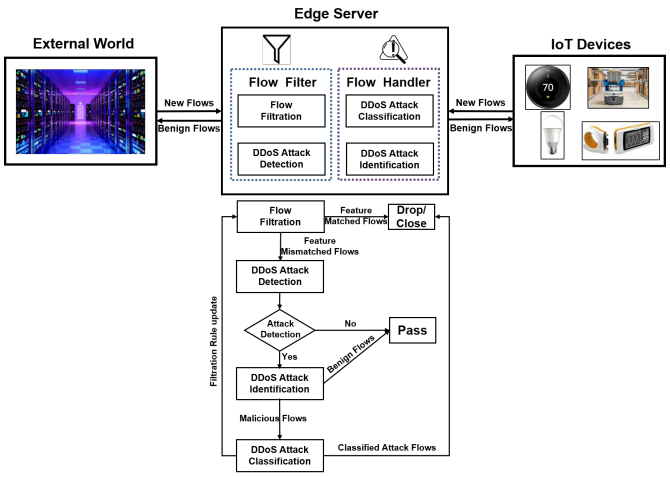


Fig. 5: Overview of FlowGuard

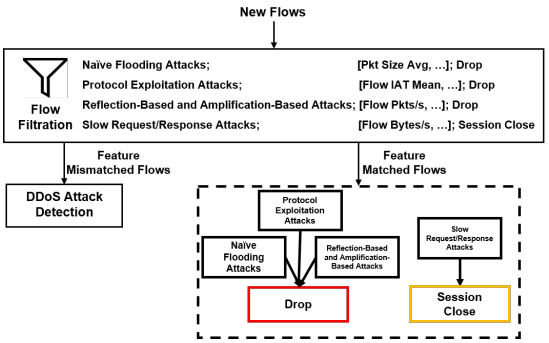


Fig. 6: The Flow Filtration Module

launched via botnets, which could be formed by compromised IoT devices as in Mirai. On the other hand, Slow Request/Response DDoS attacks can be implemented by a few IoT or other devices. Accordingly, we depict our threat model in Fig. 4, which clearly demonstrates that the two types of attack flows enter the IoT world via the edge servers. Our defense mechanism, i.e., FlowGuard, operates at the edge servers, and is responsible for DDoS attack detection, identification, classification, and mitigation.

IV. Defense Structure

In this section, we first introduce an edge-centric structure for defending against IoT DDoS attacks and mitigate their impacts on the IoT network. Then we detail the machine learning models to distinguish the malicious flows from benign ones. More specifically, we propose FlowGuard, an edge-centric IoT defense structure, aiming at providing the security chain of DDoS attack detection, identification, classification, and mitigation to defend against IoT DDoS attacks. Our structure satisfies the following design objectives:

- With FlowGuard, malicious flooding flows should be detected and processed immediately without affecting other legitimate network flows.
- With FlowGuard, even facing powerful DDoS attacks, edge servers should be able to satisfy the minimum legitimate network communication requirements and should not halt.
- FlowGuard should be efficient for normal network communications and be upgradable to dealing with zero-day DDoS attacks.

As shown in Fig. 5, FlowGuard contains two major components: a Flow Filter and a Flow Handler. The Flow Filter includes two modules, Flow Filtration and DDoS Attack Detection, which serves as an always-on security firewall with the responsibility of efficiently distinguishing the benign network flows from the malicious ones and detecting the threats of potential DDoS attacks. The Flow Handler includes two modules too, namely DDoS Attack Identification and DDoS Attack Classification, which is responsible for precisely identifying and classifying unknown DDoS attack flows. When an edge server receives incoming flows from the external world or the internal IoT devices, all the flows are first examined by the Flow Filtration module with feature comparison, and the feature-matched flows are regarded as malicious ones, which are then dropped and the corresponding open session is closed if there exists one. The feature-mismatched flows are transferred to the DDoS Attack Detection module to evaluate whether potentially some of them can cause DDoS attacks. If no threat is detected, these feature-mismatched flows pass the edge server and are sent to their destinations; otherwise, they are transmitted to the DDoS Attack Identification module, which intends to identify the benign flows from malicious ones and pass the benign flows to their destinations. The identified malicious ones are delivered to the DDoS Attack Classification module, whose responsibility is to classify the incoming flows into different attack types and handle the classified attack flows in the same way as the feature matched ones at the Flow Filtration module. At last, the DDoS Attack Classification model generates filtration rule update information and send it to the Flow Filtration module for reinforcing its filtration strength.

A. Flow Filter

In this subsection, we detail the functions of the Flow Filtration module and the DDoS Attack Detection module.

1) Flow Filtration: The Flow Filtration module makes use of a table that defines the flow filtration rules for the four IoT DDoS attack types mentioned in Section III. The table consists of four entries, with one for each DDoS attack type. Each entry contains 10 feature-value pairs to compare against for attack detection and an action field (e.g., drop the flow or close the session) to handle the attack when detected. Each entry is built from a training dataset consisting of malicious flows of the corresponding attack type as well as benigh flows. We use CICFlowMeter [28] to get 83 features for each flow, whether malicious or benign. As Information Gain (IG) is deemed to be the most effective method in measuring the entropy value of each feature when considering the features' attack relevance [29], it is employed to select the ten most attack relevant features. The value of each feature in the table is the averaged result of the feature values of all malicious flows in the training dataset. For demonstration purpose, here we list the 10 most

attack relevant features for the slow request/response DDoS attack according to IG: Flow Byts/s, Flow Pkts/s, Flow IAT Mean, Flow IAT Std, Fwd IAT Mean, Subflow Bwd Pkts, Subflow Bwd Byts, Init Bwd Win Byts, Idle Mean, and Idle Std. Note that the 10 selected features and their values for each attack type in the table are continuously updated as more malicious flows are identified.

As shown in Fig. 6, when receiving a new flow, the Flow Filtration module first extracts features from the flow and then compares the feature values with those in the filtration table. If the deviations of the ten feature values between the new flow and those of an attack type in the filtration table are all within 1 percent, we consider the new flow to be malicious and classify it to that specific DDoS attack type. The identified malicious flow is handled based on the pre-defined action in the table, i.e., either the flow is dropped or the session is closed if a session exists. The feature mismatched flows, which could be benign or malicious, are then delivered to the DDoS Attack Detection module for further process.

2) DDoS Attack Detection: Since for typical IoT tasks most IoT devices sense and upload data in a static manner, the flows passing through the edge servers, whether to or from the IoT network, should remain stable during a fixed period of time under normal conditions. Thus the abrupt increase of the real-time traffic passing through the edge servers could be caused by DDoS attacks. In order to detect the anomalous fluctuations of the flows, we develop an algorithm to observe the traffic flows going through the edge servers. According to [30], the formal notation of the traffic time series, denoted as $T(t)$ at time t, can be represented by $T(t) = P(t)+V(t)+A(t)$, where $P(t)$ represents the stable traffic rate while $V(t)$ denotes its varance (noise), and $A(t)$ denotes the anomaly caused by unknown sources. In practice, $P(t)$ and $V(t)$ are hard to obtain in realtime; thus usually estimated values, denoted by $P'(t)$ and $V'(t)$, are employed. Since the traffics to maintain the normal operations are stable, we consider the ratio of the additional real-time traffic $T(t) - P'(t)$ and the estimated stable traffic variance $V'(t)$ as a flag to signal the occurrences of anomalies. Denote this ratio by $D(t)$, i.e., $D(t) = \frac{T(t)-P'(t)}{V'(t)}$. Then DDoS attacks can be detected by comparing $D(t)$ with a threshold δ_t.

According to the anomaly flow detection idea mentioned above, we propose the following algorithm for the DDoS Detection module: when $D(t)$ is below the threshold δ_t for a certain time interval Δt, we randomly select a fraction of the flows and send them to the DDoS Attack Identification module for further analysis; otherwise, if $D(t)$ is above δ_t in Δt, we direct all flows to the DDoS Attack Identification module. Such a design is reasonable, as when $D(t)$ is large, the flows are generally malicious thus we need to send all of them to Flow Handler for attack classification and filtration rule update; while when $D(t)$ is small, the flows are largely benign but we still examine a fraction of them to be cautious enough for capturing potential attacks. Note that Δt should be chosen by considering the tradeoff between security and efficiency. Also note that such a DDoS detection approach can effectively detect DDoS attacks without interrupting the normal IoT operations.

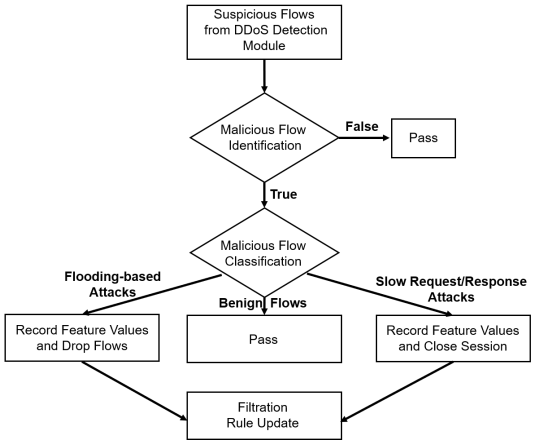


Fig. 7: The Flow Handling Process.

B. Flow Handler

1) Overview of Flow Handler: Flow Handler is the key functional unit in FlowGuard. As shown in Fig. 7, it is responsible for examining the suspicious flows coming from the DDoS Attack Detection module and identifying/classifying potential DDoS attacks. In Flow Handler, the identification of malicious flows relies on a pre-designed Long Short Term Memory (LSTM) model, which is fed with flow-oriented features such as time duration and the length of each fragment and marks them as benign or malicious. The benign flows are transmitted to their destinations while the malicious ones are sent to the Convolutional Neural Network (CNN) model for DDoS attack classification. As described in Section II, DDoS attacks are categorized into 4 classes, with three of them, namely naïve flooding attacks, protocol exploitation attacks, and reflection-based and amplification-based attacks, being flooding-based, and the fourth one referring to the slow request/response attacks. Our CNN model is able to classify the flows into benign ones or one of the four types of DDoS attacks. Since flooding-based attacks and slow request/response attacks are different in nature, their defense strategies are different, i.e., either dropping the flows or closing the session if a session exists, as shown in Fig. 6. After malicious flows are classified, their features are used to update the filtration rules.

The reason for adopting an LSTM in the identification stage is because as a connectionist model, LSTM is good at capturing the dynamics of the flows via cycles and selectively retaining the information across sequence steps by feeding one element at a time, and DDoS attack traffic is temporally correlated. On the other hand, in the classification stage, we employ a Convolutional Neural Network (CNN) because it has strong abilities to cope with classification issues by varying the depth and breadth of the network as demonstrated by the promising results in image classification, audio classification, video classification, sentence type classification, and sentence classification [1] [31] [32] [33] [34]. CNN can cope with time-sensitive attack situations with fewer connections and

parameters compared to the standard feed-forward neural networks with similar numbers of layers. Additionally, it can utilize the property of many natural signals with compositional hierarchies, in which higher-level features are obtained by composing lower-level ones. This property is particularly suitable for the classification of DDoS attacks because each type of attacks has a unique set of features.

Note that in practice, the flows entering the DDoS Identification module are largely benign ones, which could significantly affect the classification performance of CNN if no LSTM is adopted. We employ LSTM's nice feature of capturing the temporal dynamics to identify most benign flows such that the following CNN can do a better job of classification. Also note that the constructions of LSTM and CNN need to consider both accuracy and time efficiency, which are detailed in the following two subsections.

2) DDoS Identification Module: The DDoS Identification Module implements an LSTM network with m layers: an input layer followed by m-2 hidden layers followed by an output layer. The input layer contains k neurons, where k is the number of features from the dataset. We resize the data into a 2-D vector with shape (n, k) before the data is fed into the input layer, where n is the batch size or the number of flows. Its output is reshaped into a 3-D vector with the shape (n, n_step, n_input) as the input to the first hidden layer, where n_step is the number of times the features are fed into the hidden layer and n_input is the number of features being fed each time. Following the hidden layers is a BatchNormalization function, which is used to ensure that the output of the hidden layers is not over-fitting. Finally, the normalized data is input to the output layer possessing a softmax function, whose outputs are used to distinguish malicious flows from benign ones. Finally, LSTM outputs the features with the shape (2,) for the DDoS Classification Module.

For better elaboration, we denote the data output by the i-th layer of our LSTM model to be X_i^l, where i = 1, 2, 3, $\cdots$, the weight vector between the i-th layer and the i+1-th layer to be $W_{i,i+1}^l$, and the bias vector of the i-th layer to be B_i^l. Then one can represent the relationship between the data from two neighboring layers as:

$$X_{i+1}^l = \frac{e^X - e^{-X}}{e^X + e^{-X}} \tag{1}$$

where the relationship is a tanh. Having had a big picture of the LSTM structure, our next step is to take a closer look at it. Let x_t be a data vector fed into a LSTM cell at time t, h_t be the output by the cell, f_i be the input gate, f_f be the forget gate, f_o be the output gate, and C_t be the cell state [35]. Then within an LSTM cell, the following calculations are performed:

$$\mathrm{f}_i = \delta(W_I \cdot h_{t-1} + W_I \cdot x_t) + B_I \tag{2}$$

$$\widetilde{C_t} = \tanh(W_C \cdot h_{t-1} + W_C \cdot x_t + B_C) \tag{3}$$

$$f_f = \delta(W_F \cdot h_{t-1} + W_F \cdot x_t) + B_F \tag{4}$$

$$C_t = f_f * C_{t-1} + f_f * \widetilde{C_t} \tag{5}$$

$$f_o = \delta(W_O * h_{t-1} + W_o \cdot x_t) + B_o \tag{6}$$

$$h_t = f_o * \tanh(C_t) \tag{7}$$

where $\widetilde{C_t}$ is an intermediate value between C_t and C_{t-1}. The variables with letters W and B refer to the weights and biases, respectively, for that specific gate or cell state, and σ represents the sigmoid function with the form of

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{8}$$

3) DDoS Classification Module: Following the DDoS Identification Module is the DDoS Classification Mdoule that employs a CNN algorithm for attack classification. Our CNN has 3 types of layers: convolutional layers, max-pooling layers, and fully connected feed-forward layers. The data output by the i-th layer is denoted to be X_i^c, where i = 1, 2, 3, $\cdots$, the weight vector between the i-th layer and the i+1-th layer is denoted by $W_{i,i+1}^c$, and the bias vector of the i-th layer is denoted by B_i^c. Similar to LSTM, our data is first resized into a 3-D vector with the shape of (n, n_step, n_input), where n is the number of samples being fed into the network. Then we reshape each data with shape (n_step, n_input) to (n_step, n_input, 1) via flattening and perform a normalization before sending it to the first convolutional layer. The first convolutional layer contains nf zero-padded filters with a size of (fs, fs). The activation function used here is again Sigmoid. Then one can get the output with one more dimension in depth, i.e., X_1^c has the shape of (n, n_step, n_input, nf). A max-pooling layer follows the first convolutional layer with the stride of st and the pooling size of (ps, ps). X_2^c thereby has the shape of (n, $\lceil n\_step/ps \rceil$, $\lceil n\_input/ps \rceil$, nf).

Following the first max-pooling layer we have a convolutional layer then a max-pooling layer, and such a structure is repeated for a number of times. The output of the last max-pooling layer is fed into the fully connected feed-forward layers which finally output the features with a shape of (fl), where fl is the number of attack classes the dataset has.

V. Evaluation

In this section, we present our evaluation procedure and demonstrate the results. Since the Flow Filtration module follows a preliminary filtration procedure with strict rule restrictions, and the DDoS Attack Detection module is a real-time flow composition assessment, they have limited impacts on the performance of FlowGuard. Thus, in this section, we focus on the performance of the Flow Identification and Flow Classification modules to assess the effectiveness of FlowGuard.

Attack Types	Total Packets	Total Flows
ICMP Flooding Attacks	1,000,011	999,974
UDP Flooding Attacks	1,000,039	999,997
TCP Flooding Attacks	1,000,061	1,000,000
Slow Headers Attacks	1,003,507	21,087
Slow Read Attacks	1,000,109	22,015
Slow Write Attacks	1,000,085	21,137
Slow Body Attacks	1,002,851	21,535

TABLE I: The Number of Captured Malicious Packets and the Number of Corresponding Flows for the 7 Attack Types

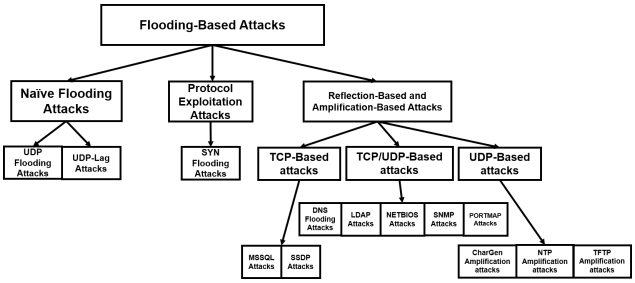


Fig. 8: The Composition of the DDoS Dataset

A. Experimental Setup

We created a testbed to simulate a real attack scenario in a personal computer based edge server. Since FlowGuard is adopted at the edge server, a PC fulfills our experimental setup requirements for the performance evaluation purpose. The PC employed by our experiment is equipped with 12 processors, 6 kernels, and an installed RAM with a 32.0GB available memory running 64-bit Windows 10 operating system. Each processor is configured with Intel(R) Core(TM) i7-8750H CPU @2.20GHz, 2201 Mhz. The versions of Keras and Tensorflow to develop our models are 2.2.3 and 1.14.0, respectively. Besides, to obtain a properly sized dataset, we set up an attacker VM and a victim VM on our PC.

Datasets. In our experiment, the attacker VM used BoNeSi and SlowHTTPTest to launch different types of attacks at a rate of 10,000 packets per second (PPS) to deplete the resources at the victim VM. We recorded all the traffics between the two VMs and collected 7 attack datasets, with each having over 1,000,000 malicious packets which were further merged into flows by the Wireshark traffic capturing tool. The flows were then sent to CICFlowMeter [28] to extract 83 features for each of them. Note that BoNeSi is a DDoS botnet simulator that is capable of generating ICMP, UDP, and TCP flooding attacks with a flexible size of a botnet [36]; SlowHTTPTest is a DDoS slow attack simulation tool [37] used to initiate slow headers, slow read, slow write, and slow body attacks to act as slow request/response attackers in our experiment; and CICFlowMeter is a flow-based packet feature extractor. The detailed information of the captured flows is shown in Table I. We have one dataset for each attack type supported by BoNeSi and SlowHTTPTest.

Besides the attack traffics generated by the simulators, we adopted CICDDoS2019 [38] [39], a dataset provided by the Canadian Institute for Cybersecurity (CIC) and University of New Brunswick (UNB). These two organizations distribute famous intrusion and DDoS attack datasets once per year and those datasets are used by many security studies [40] [41]. CICDDoS2019 includes more than 1 million benign flows and over 30 million malicious ones, which include 13 classes of modern DDoS attacks such as NTP, DNS, LDAP, MSSQL, and NetBIOS. The composition of this dataset is shown in Fig. 8.

Model Parameters. In order to compare with the state-of-the-art works, we studied the performance of our LSTM and CNN models over CICDDoS2019. We also applied the LSTM and CNN models trained by the CICDDoS2019 dataset to the dataset generated by the simulation tools mentioned above to validate the generalization ability of these two models. To identify a proper structure for achieving our goals, we attempted the number of layers for both LSTM and CNN from 1 to 10, and it turned out that structures with 3 layers for the LSTM and 6 layers for the CNN are the most effective ones because they can achieve the highest accuracy with a reasonable efficiency. More specifically, our LSTM model includes one fully connected feed-forward layer as the input layer, one layer of a Long Short-Term Memory (LSTM) cell as the hidden layer, and one fully connected feed-forward layer as the output layer, in which n, n_step, n_input, and k are set to 1280, 4, 10, and 40, respectively. The CNN includes 2 convolutional layers, 2 max-pooling layers and 2 fully connected feed-forward layers, in which the parameters n, n_step, n_input, fs, st, ps, and fl are set to 128, 2, 40, 5, 2, 2, and 12, respectively, while nf is set to 32 and 64 for the first and second convolutional layers, respectively.

Performance Metrics. The metrics to measure the performance of LSTM and CNN modules are accuracy and efficiency. Specifically, we adopt four common machine learning evaluation metrics to examine our model accuracy:

- Accuracy: Accuracy is the ratio of correctly predicted malicious flows (TP) and benign flows (TN) over all the flows (TP+TN+FP+FN).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{9}$$

- Precision: Precision is the ratio of correctly identified malicious flows over all predicted malicious flows (TP+FP).

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

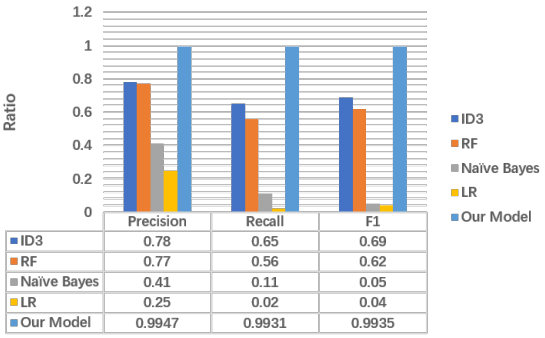
- Recall: Recall is the ratio of correctly predicted malicious flows over the total malicious flows (TP+FN).

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

- F1 Score: F1 Score is the balanced average of precision and recall.

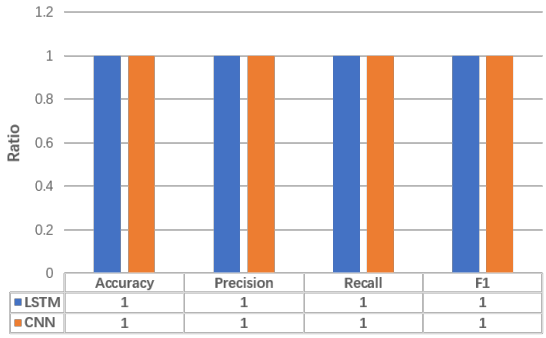
$$F1 = \frac{2 * (Recall * Precision)}{Recall + Precision} \tag{12}$$

Besides the evaluation on accuracy, we also examine the efficiency of the DDoS attack identification and classification modules. Since IoT is a time-sensitive system, the efficiency of these two modules can be measured by the CPU processing



	Precision	Recall	F1
ID3	0.78	0.65	0.69
RF	0.77	0.56	0.62
Naïve Bayes	0.41	0.11	0.05
LR	0.25	0.02	0.04
Our Model	0.9947	0.9931	0.9935

Fig. 9: The Accuracy Evaluation Results of Five Models with CICDDoS2019



	Accuracy	Precision	Recall	F1
LSTM	1	1	1	1
CNN	1	1	1	1

Fig. 10: The Accuracy Evaluation Results of LSTM and CNN Models with the Simulated Dataset

time per flow. Let $CPUTime$ be the total clock ticks used to process $TestSets$ number of flows. We have

$$ModuleProcessEfficiency(MPE) = \frac{CPUTime}{TestSets} \quad (13)$$

The lower the MPE, the more number of flows the identification and classification modules can process within a certain time period. An ideal defense structure should not cause a significant delay of the normal operations.

B. Evaluation Results

1) Accuracy Evaluation with Dataset CICDDoS2019: We first evaluated the accuracy of our LSTM model with the CICDDoS2019 dataset, which contains 83 features for each flow. The universal features such as the global average data and the number of session initialization packets, can significantly affect the malicious flow identification accuracy as they do not carry discriminatory information, thus we made use of 40 relevant features that correspond to malicious flow identification in our experiment: we selected 7 features that can mark a specific flow (such as Flow.ID, Source.IP, and Destination.IP) to locate the identified malicious flows, 15 features that can represent the overall characteristics (such as the Total Length of Bwd Packets and the Total Length of Fwd Packets) to evaluate the flow contents, and 18 time-related features that could be used for the attack identification (such as the Flow IAT Std, which is the standard inter-arrival time in the forward direction, and the Bwd IAT Std, which is the standard inter-arrival time in the backward direction) to help our model distinguish malicious flows from benign ones. We also compared the identification accuracy of our LSTM model against the 4 machine learning models (ID3, Random Forest, Nave Bayes, and logistic regression) proposed in [39], which also employed CICDDoS2019 for performance validation.

We conducted multiple simulation trials with training datasets ranging from 2 million flows to 30 million flows, and randomly chose a testing dataset with a volume of 100 thousand flows from over 1 million testing flows. The results indicate that our LSTM model has an average accuracy of 98.9%, which far outperforms the contrastive machine learning models whose highest accuracy is 78%. The detailed comparison results are shown in Fig. 9.

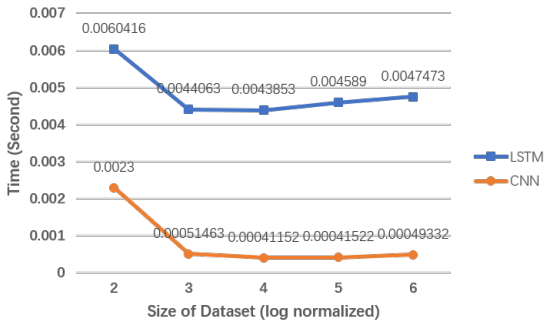


Fig. 11: The Efficiency Evaluation Results of LSTM and CNN Models

Since there does not exist any study employing the CICDDoS2019 dataset for flow classification purpose based on our best knowledge, we focus on the classification accuracy of our CNN model without comparing with any other model. In this study, we selected 80 features, and excluded 3 features (RST Flag Cnt, PSH Flag Cnt, and ECE Flag Count) that have no impact on the DDoS attack classification accuracy. We then trained and tested our CNN model with different data from CICDDoS2019. The results indicate that the accuracy of our CNN model in distinguishing each type of malicious flows from the other 4 DDoS attack types as well as the benign flow category is up to 99.9%.

2) Generalization of Our Model: In this subsection, we evaluate the generalization of our models with the labeled data collected from BoNeSi. The dataset is composed of 3 million labeled DDoS attack flows (1 million flows for each of UDP, ICMP, and TCP DDoS attacks) and 100 thousand labeled benign ones. The results are shown in Fig. 10. One can see that our models can perfectly identify and classify all malicious flows.

Since CICDDoS2019 does not contain the slow request/response attacks, we retrained our LSTM and CNN models with the combination of the CICDDoS2019 dataset and 28 thousand slow attack flows (7 thousand flows for each type of the slow attacks collected from SlowHTTPTest). Then we evaluated our models with the test dataset which consists of 3 million labeled DDoS attack flows generated by BoNeSi, 56 thousand labeled slow attack flows created by SlowHTTPTest,

and 100 thousand labeled benign traffic flows. The values of accuracy, precision, recall and F1 stay at 100%.

3) Evaluation on Efficiency.: Since IoT is a time-sensitive system, the procedure for distinguishing attacks should not cause a significant delay. In this study, we measured the CPU time of different magnitudes of datasets passing through our models during all testing procedures, and calculated the average MPE per flow. According to the efficiency evaluation results shown in Fig. 11, one can conclude that the processing time of LSTM for each flow is between 0.004 seconds and 0.047 seconds, while the processing time of CNN for each flow is within millisecond. The efficiency of LSTM may not perfectly satisfy the delay requirement of IoT. We attribute this to the fact that our experiments were carried out on a personal computer that is not equipped with powerful processing CPUs and high-speed memories. As reported by [42], the internal constraints of a PC would tremendously affect the processing time and make the efficiency lower than a computer with a GPU at least 10 times in the same testing environment. Thus one can safely claim that if our models are running on an edge server with more powerful computational resources than a PC, the normal IoT operations would not be obviously affected by our LSTM and CNN models.

VI. Conclusion

In this paper, we proposed a novel IoT DDoS defense scheme titled FlowGuard and constructed two novel machine learning models for DDoS identification and classification. We first detailed the design of FlowGuard, which consists of two components, namely Flow Filter and Flow Handler. Flow Filter is in charge of filtering malicious flows based on filtration rules established by the Flow Handler, and detecting unidentified malicious flows based on traffic variations. Flow Handler takes the responsibility of identifying and classifying malicious flows according to two machine learning models LSTM and CNN developed in this paper. The performances of these two models were extensively studied based on a simulated dataset and the CICDDoS2019 dataset.

Acknowledgment

This work was partially supported by the National Key R&D Program of China under grant 2019YFB2102302, the National Natural Science Foundation of China under grants 61971014, 11675199, 61832012, and 61771289, the National Science Foundation of the US under Grants IIS-1741279 and CNS-1704397, and the Deanship of Scientific Research at Prince Sattam Bin Abdulaziz University under the research project No. 2019/01/10411.

References

[1] "Size of the internet of things (iot) market worldwide from 2017 to 2025," https://www.statista.com/statistics/976313/global-iot-market-size/l, 2018.

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[3] "Inside the infamous mirai iot botnet: A retrospective analysis," https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/, 2017.

[4] "The iot rundown for 2020: Stats, risks, and solutions," https://securitytoday.com/Articles/2020/01/13/The-IoT-Rundown-for-2020.aspx?Page=2, 2020.

[5] "More than half of iot devices vulnerable to severe attacks," https://threatpost.com/half-iot-devices-vulnerable-severe-attacks/153609/, 2020.

[6] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE communications surveys & tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[7] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, 2004.

[8] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[9] K. Sonar and H. Upadhyay, "A survey: Ddos attack on internet of things," *International Journal of Engineering Research and Development*, vol. 10, no. 11, pp. 58–63, 2014.

[10] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.

[11] A. Belenky, "Ip traceback with deterministic packet marking dpm," https://digitalcommons.njit.edu/cgi/viewcontent.cgi?article=1641& context=dissertations, 2003.

[12] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source." in *LISA*, 2000, pp. 319–327.

[13] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-based ip traceback," in *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4. ACM, 2001, pp. 3–14.

[14] T. Chen, R. Xu, Y. He, and X. Wang, "Ip traceback-based intelligent packet filtering: a novel technique for defending against internet ddos attacks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 9, 2003.

[15] G. Nychis, V. Sekar, D. Andersen, H. Kim, and H. Zhang, "An empirical evaluation of entropy-based traffic anomaly detection," *Proceedings of the 8th ACM SIGCOMM conference on Internet Measurement*, 2008.

[16] D. Boro and D. Bhattacharyya, "Dyprosd: a dynamic protocol specific defense for high-rate ddos flooding attacks," *Microsystem Technologies*, vol. 23, 2017.

[17] S. Oshima, T. Nakashima, and T. Sueyoshi, "Early dos/ddos detection method using short-term statistics," *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, Feb. 2010.

[18] S. Roschke, F. Cheng, and C. Meinel, "Intrusion detection in the cloud," in *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 2009, pp. 729–734.

[19] P. Rengaraju, V. R. Ramanan, and C.-H. Lung, "Detection and prevention of dos attacks in software-defined cloud networks," *2017 IEEE Conference on Dependable and Secure Computing*, Aug. 2017.

[20] P. Xiao, W. Qu, H. Qi, and Z. Li, "Detecting ddos attacks against data center with correlation analysis," *Computer Communications*, vol. 67, pp. 66–74, 2015.

[21] C. She, W. Wen, Z. Lin, and K. Zheng, "Application-layer ddos detection based on a one-class support vector machine," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 9, no. 1, pp. 13–24, January. 2017.

[22] R. Vishwakarma and A. K. Jain, "A honeypot with machine learning based detection framework for defending iot based botnet ddos attacks," *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, April. 2019.

[23] M. Asad, M. Asim, T. Javed, M. O. Beg, H. Mujtaba, and S. Abbas, "Deepdetect: Detection of distributed denial of service attacks using deep learning," *The Computer Journal*, 2019.

[24] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in iot networks," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0452–0457, 2019.

[25] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," *2018 IEEE Security and Privacy Workshops (SPW)*, 2018.

[26] C. She, W. Wen, Z. Lin, and K. Zheng, "Dad-mcnn: Ddos attack detection via multi-channel cnn," *Proceedings of the 2019 11th Interna-*

tional Conference on Machine Learning and Computing, pp. 484–488, February. 2019.
[27] PNNL, "Edge computing," *White paper*, Jan. 2013.
[28] "Cicflowmeter," https://github.com/ahlashkari/CICFlowMeter, 2019.
[29] O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for ddos detection in cloud computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, p. 130, 2016.
[30] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang, "Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 91–92.
[31] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, and etc., "Cnn architectures for large-scale audio classification," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March. 2017.
[32] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2285–2294, 2016.
[33] S. Zha, F. Luisier, A. Walter, N. Srivastava, and R. Salakhutdinov, "Exploiting image-trained cnn architectures for unconstrained video classification," https://arxiv.org/abs/1503.04144, 2015.
[34] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan, "Hcp: A flexible cnn framework for multi-label image classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1901–1907, 2015.
[35] Y. Xiao, Y. Jia, X. Cheng, J. Yu, Z. Liang, and Z. Tian, "I can see your brain: Investigating home-use electroencephalography system security," *IEEE Internet of Things Journal*, vol. 6, no. 4, 2019.
[36] "Bonesi," https://github.com/Markus-Go/bonesi, 2018.
[37] "Slowhttptest," https://github.com/shekyan/slowhttptest, 2019.
[38] "Ddos evaluation dataset," https://www.unb.ca/cic/datasets/ddos-2019.html, 2019.
[39] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–8.
[40] V. Kanimozhi and T. P. Jacob, "Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing," in *2019 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2019, pp. 0033–0036.
[41] Q. Zhou and D. Pezaros, "Evaluation of machine learning classifiers for zero-day intrusion detection–an analysis on cic-aws-2018 dataset," *arXiv preprint arXiv:1905.03685*, 2019.
[42] "Gpus vs cpus for deployment of deep learning models," https://azure.microsoft.com/en-us/blog/gpus-vs-cpus-for-deployment-of-deep-learning-models/, 2018.