Computer Networks 209 (2022) 108874

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

SmartDefense: A distributed deep defense against DDoS attacks with edge computing

Sowmya Myneni [a,*], Ankur Chowdhary [a], Dijiang Huang [a], Adel Alshamrani [b]

[a] Arizona State University, 699 S Mill Ave, Tempe, 85281, AZ, USA
[b] University of Jeddah, Jeddah, Saudi Arabia

ARTICLE INFO

Keywords:
Distributed Denial of Service (DDoS)
Internet of Things (IoT)
Botnets
Edge computing
Deep neural networks

ABSTRACT

The growing number of IoT edge devices have inflicted a change in the cyber-attack space. The DDoS attacks, in particular, have significantly increased in magnitude and intensity. Of the existing DDoS solutions, while the destination-based defense mechanisms incur high false positives due to the seemingly legitimate nature of the attack traffic, defense mechanisms implemented at the source alone do not suffice due to the lack of visibility into ongoing DDoS attacks. This paper proposes a distributed DDoS detection and mitigation framework, SmartDefense, based on edge computing approaches towards detecting and mitigating DDoS attacks at and near the source. By mitigating the DDoS attacks near the source, SmartDefense significantly reduces unnecessary bandwidth otherwise consumed by DDoS traffic going from residential edge networks to the ISP edge network.

Furthermore, SmartDefense demonstrates how ISPs can detect botnet devices in their customer's network by having smart edge devices pass attributes that are processed by the botnet detection engine at the provider's edge. The evaluation of this work shows that SmartDefense can improve the detection and mitigation rate, with over 90% of DDoS traffic caught at the source and over 97.5% of remaining DDoS traffic caught at the provider's edge. Our experiments also demonstrate how using a botnet detection engine can further reduce the DDoS traffic by up to 51.95% by facilitating ISPs to detect bot devices in their customers' edge network.

1. Introduction

A Distributed Denial of Service (DDoS) attacks on *Dyn* and *Krebs on Security* websites in 2016 have been their first of kind alerting researchers and industry professionals alike, of the potential threat posed by edge-based IoT devices. The 1.2 Tbps attack on *Dyn* [1] has marked the beginning of the Terabit attack era. Each year since then has seen DDoS attacks with increased intensity, e.g., *GitHub* in 2018 with 1.35 Tbps of data and *126.9 million packets per second* (Mpps) [2], 580 Mpps DDoS attack in 2019 [3], DDoS attack on *AWS* in 2020 with a 2.3 Tbps of data [4]. The growing number of IoT devices and the increasing bandwidth and intensity of the traffic are continuously challenging the Internet Service Providers (ISPs) in providing continuous fast delivery and securing themselves against the attack itself.

The current research primarily consists of two types of defense mechanisms - (1) destination-based defense; and (2) source-based defense approaches. While destination-based defenses involving traditional rate limiting, packet filtering, along with machine learning techniques that new solutions have embraced, have become prominent with evolving cloud computing, they alone do not suffice [5,6]. The intensity at which the current DDoS attacks [7] are happening can often give no option to the defender at the destination but to eventually take the target system off the Internet or transform the threat into an Economic Denial of Sustainability (EDoS) attack [8] on the target that is coping up with the attack by scaling the resources. Even popular commercial products that offer DDoS protection, such as Corero, AWS Shield, Cloudflare, etc., are challenged by the increasing volume of attacks that last for days. On the other hand, source-based mitigation solutions are capable of preventing malicious traffic from getting onto the Internet [9,10] and thus reduce unnecessary usage of Internet resources. However, they alone fail to be effective, specifically in the case of volumetric DDoS attacks, as the distributed nature of these DDoS attacks makes the edge devices oblivious to the huge volume that passes through the ISP's edge gateways. ISPs, bridging the source and destination, are better positioned to handle attacks like DDoS.

Existing solutions for ISPs to defend against DDoS attacks involving statistical and traditional machine learning-based approaches [11–13]

* Corresponding author.
E-mail addresses: sowmya.myneni@asu.edu (S. Myneni), achaud16@asu.edu (A. Chowdhary), dijiang.huang@asu.edu (D. Huang), asalshamrani@uj.edu.sa (A. Alshamrani).

https://doi.org/10.1016/j.comnet.2022.108874
Received 20 October 2021; Received in revised form 3 February 2022; Accepted 24 February 2022
Available online 16 March 2022

can be easily overwhelmed with current rates of DDoS traffic. Software-defined networking (SDN) based networks have been adopted as an industry standard to manage large-scale networks. A distributed SDN framework, as discussed by Bannour et al. [14] can be employed to address the scalability challenges that are inherent in secured management of emerging technologies such as the Internet of Things (IoT). SDN-based anomaly detection systems have been used to defend against port-scanning and DDoS attacks [15]. We consider a deep-learning-based anomaly detection mechanism in this work to detect DDoS attacks using traffic information from customer edge and provider edge in the network.

While deep learning techniques could address the issues associated with huge volumes of traffic, existing solutions [16] either use machine learning techniques such as Self-Organizing Map (SOM) that are slow in training speed or often result in high false positives on real-time traffic. Solutions that use deep reinforcement techniques [17,18], can learn the traffic pattern for attacks with sufficient data and well-defined features. However, deep-reinforcement learning techniques are often unstable when trying to find an optimal solution (attack pattern) and thus could lead to inconsistencies and delays in processing requests by ISPs. One key to mitigating DDoS attacks is identifying the sources that make the attack happen and blocking the traffic at the ISP's edge network.

To achieve this goal, in this work, we present *SmartDefense* that relies on a two-stage DDoS detection strategy running on both customer edge (CE) and provider edge (PE). This approach leverages edge computing capabilities to allow ISPs to detect and mitigate DDoS attacks at and near the source while also facilitating the identification of botnet devices in their customers' networks. The CE and PE are equipped with smart programmable devices (CER and PER respectively) with upgraded processing power that can run DDoS defense module, C-Defense at CE, and P-Defense at PE, respectively, to identify and block the outgoing DDoS traffic. The C-Defense component within CER runs DDoS detection based on a Deep Neural Network (DNN) approach that is regularly trained and updated by the ISP. Attack traffic detected as DDoS attack vectors with high-prediction accuracy are dropped at the CE based on the threshold configured by the ISP, with all the other residential network-generated traffic forwarded to the PE. At the PE, P-Defense further processes the aggregated residential traffic from CEs with and without the C-Defense component. While all the traffic coming from a CER without the C-Defense component is processed at the PE's detection engine, the traffic coming from CER equipped with the C-Defense component is evaluated at PE (on CE's probability predictions P_p^e) to determine the need for further processing. Traffic coming from a C-Defense component carrying a prediction probability greater than a threshold T_e will only be further processed at the PE's detection engine. Thus, the ISP edge system will consume smart residential edges with more implemented and less DDoS detection processing overhead.

The results of the proposed two-stage edge-computing approach prove to be a promising solution for ISPs to reduce unnecessary bandwidth usage and latency due to otherwise accumulated malicious traffic at the provider edge. Another crucial factor for ISPs is Consistency. Achieving consistency during peak hours and during an ongoing DDoS attack is challenging for ISPs. With the proposed hierarchical approach, *SmartDefese*, ISPs can ensure consistency to their subscribers even during peak hours or during an ongoing DDoS attack as the processing load is distributed across the customer edge routers. The ability of botnet engine to detect bots and the use of that information to drop highly-probable malicious traffic originating from the bot devices rather than cause latencies in processing the legitimate traffic is an added incentive to ISPs using *SmartDefense*. While the research work in this paper portrays the approach to defend against DDoS attacks, the approach is not limited and can be extended to other issues that ISPs are facing such as Spam etc. On this note, we present case studies based on two DNN models: one lightweight Feed-forward Neural Network (FNN) and another Long-Short Term Memory (LSTM), a Recurrent Neural Network (RNN) pointing out how even a simple neural network can benefit ISP when using the proposed system. The key contributions of this research work are as follows:

- Reducing ISP Overhead: We present a two-stage edge computing-based DDoS detection framework that significantly reduces the overhead in processing DDoS traffic coming into the ISP's edge network. Through an emulation of a residential edge and the ISP's edge, we show how our proposed approach can mitigate over 90% of DDoS traffic originating from the customer edge.
- Improved Accuracy: Utilizing the attributes passed by the customer's edge network, the ISP's at their edge can improve the overall accuracy on detecting and mitigating DDoS traffic. Our experiments demonstrate that over 97.5% of the DDoS traffic that failed to be detected at customer edge can be mitigated at provider edge.
- Detecting Bots: We further present how ISPs can detect botnet devices using a botnet detection engine that utilizes the attributes sent by customers' edge devices. Our evaluation shows that the use of botnet detection engine can further reduce the DDoS traffic, by up to 51.95%, which otherwise is sent to the ISP outward gateway towards the destination.

The rest of the paper is arranged as follows: in Section 2, we present the related work; Section 3 presents the system models to build the foundation of this work; Section 4 presents the details of the proposed algorithms on customer's edge and provider's edge; the performance evaluation and case studies are presented in Section 5; and finally, we discuss future work and conclude our work in Section 6.

2. Related work

DDoS attacks find themselves facing several mitigating solutions. However, the increasing bandwidth and the intensity of these attacks with the increase in interconnected devices call for new approaches to mitigating the attacks. In this section, we present some of those solutions related to our research in this paper (see Table 1).

Argyraki and Cheriton [19] proposed AITF protocol that, when deployed at an edge, the victim site, helps in filtering the DDoS traffic coming from another edge, the attacker source, implementing the AITF protocol. While this solution offers efficient filtering capabilities, it is limited to filtering DDoS traffic generated using the source address spoofing technique. It thus fails to detect and mitigate the seemingly legitimate DDoS traffic that originates from bot devices. Mirkovic and Reiher [9] proposed D-WARD, a source-based defense solution, against flooding attacks through filtering and rate-limiting of suspicious flows near the source level. Chen and Song [20] proposed two perimeter-based defense mechanisms for ISPs to provide the anti-DDoS service. This solution protects the victim and still does not address the problem of unwanted traffic flowing to ISP's core network from its edge networks and customers.

Chen et al. [21] have proposed a solution for ISPs using change aggregation trees to detect and alert as flooding occurs, followed by dropping of suspicious packets or rate-limiting the flows. This solution helps ISPs in detecting DDoS traffic. Still the lack of incentives to the ISP in deploying this solution makes it less adoptable. Stewart et al. [10] proposed another source based defense approach where in a guardian node between the home router and modem looks at the traffic flowing in and out of the home network towards stopping DDoS attacks originating from legal nodes that got compromised by malware. The drawback of this solution is that it requires the guardian nodes to be setup on each home network, which is not feasible.

Bhardwaj et al. [24] have proposed ShadowNet architecture which involves deploying edge functions on the distributed edge infrastructure. These edge functions sketch the profiles of the IoT traffic from a given location through them and send them in the form of shadow-packets to a ShadowNet web service hosted in the cloud for aggregated analysis. While the solutions seems feasible theoretically, it does raise serious practical concerns. The number of shadow packets generated per request could quickly overwhelm the edge, making it a victim of a

Table 1
A summary of research works addressing DDoS attacks.

Research work	Technique used	Details
[19], 2005	AITF filter	Filters DDoS traffic generated using source address spoofing technique.
[20], 2005	Perimeter defense	Edge routers for collectively identifying and rate-limiting the flooding sources.
[9], 2005	D-WARD	Source based defense solution, against flooding attacks through filtering and rate limiting.
[21], 2007	Aggregation trees	Change aggregation trees for attack detection with rate-limiting for preventing DDoS.
[22], 2016	Deep learning	Deep learning based DDoS detection.
[10], 2017	Guardian node	Source based DDoS defense approach where in a guardian node between the home router and modem.
[23], 2018	Edge-based defense	Congestion detection and real-time traffic monitoring.
[24], 2018	ShadowNet	Edge function based traffic profiling and aggregated analysis.
[25], 2018	MECPASS	Source level filtering at edge layer, and network level filtering in the cloud.
[26], 2019	CNN	Flow feature extraction with autoencoder.
[27], 2019	Supervised ML	Traffic anomaly detection using supervised machine learning.
[28], 2019	LSTM	Deep learning based DDoS detection.
[18], 2020	DCNN	Edge computing with Q-Learning.
[29], 2020	DDoS edge defense	Two machine learning models for DDoS identification and classification.
[30], 2021	CODE4MEC	DDoS defense for mobile edge computing that adapts to traffic changes.
[31], 2021	Access control	Attribute-bases access control (ABAC) for ensuring data privacy.
[32], 2021	MECShield	Mobile-edge based DDoS attacks detection using multiple smart filters.
[33], 2021	FUPE	Security task scheduling for SDN-based defense against TCP SYN-Flood attacks.

DDoS attack. Dao et al. [23] have proposed a novel solution that leverages edge computing in 5G networks to prevent DDoS attacks at the source through congestion detection and real-time traffic monitoring. However, the proposed solution is based on a client–server architecture with a client deployed at the edge computing layer and server at the core network making it less feasible of a solution.

Hwang et al. [26] have proposed a solution for edge servers to detect botnet originated DDoS attacks. Their proposed solution uses a convolutional neural network (CNN) to extract the features of a flow, which are then passed to an autoencoder that is trained on normal traffic towards detecting malicious traffic. While the approach detects malicious traffic through anomaly detection, the solution needs to be implemented on edge servers with high processing and memory capabilities, limiting its implementation to IoT gateways and not residential networks that contribute to bot traffic. Nguyen et al. [25] have proposed a novel approach, MECPASS, that is capable of filtering DDoS attack traffic at source and network-level in Mobile Edge Computing (MEC) environment. The source level filtering happens at the MEC servers in the edge layer, while the network level filtering occurs at the MEC servers in the cloud. Though the false positives are contained within 1% through this approach, the solution requires MEC servers at the source-level to scale well on a large network. ISP-based solutions have a significant impact on effectively controlling volumetric DDoS attacks. However, unwanted attack traffic still flows to ISP, leading to unnecessary resource consumption. ISP solutions can coalesce with source-based defense techniques and can help in effectively addressing DDoS attacks.

Khosroshahi and Ozdemir [27] proposed a solution to detect sources that are being used for DDoS attacks. Their proposed model intercepts the outgoing traffic from Android devices and evaluates it against the user's previously captured and obtained normal and abnormal behavior using supervised machine learning techniques. While the authors claim the solution detects anomalous behavior, using supervised machine learning techniques will only capture the known attack behavior and thus not true anomalous behavior.

Flowguard [29] deals with DDoS attacks against IoT devices. It utilizes two machine learning models for DDoS identification and classification. The authors used the CICDDoS2019 data set to benchmark Flowguard. CPSS [18] provides edge-defense against low-rate DDoS attacks and utilizes deep-learning model DCNN based on Q-network as the decision-maker. This solution is well suited for edge defense. We utilize an architecture-aware DDoS defense that helps identify the attack pattern using information from both core and edge-network resources.

Han et al. [31] proposed blockchain-based access control for data privacy in IoT environments. The solution utilizes attribute-based access control to ensure data privacy. The approach may help ensure confidentiality and integrity of the critical data, and it can work well with SmartDefense to provide data availability. CODE4MEC [30] provides a defense for mobile edge computing that adapts to traffic changes. The solution uses four control-plane functions to provide life cycle management to deal with application-layer DDoS attacks. SmartDefense provides defense against various DDoS attacks targeting network and application layers. FUPE [33] provides security task-aware defense for IoT devices. FUPE utilizes fuzzy-based multi-objective particle swarm optimization to aggregate optimal computing resources and the information for resource-aware security protection. The research work deals with TCP SYN-Flood DDoS attacks, and it may be challenging to adapt to different variants of DDoS attacks using the FUPE approach.

Dao et al. [32] have proposed MECShield, a localized DDoS prevention framework leveraging MEC capabilities to detect and prevent DDoS attacks by deploying multiple smart filters at the edge of relevant source/destination networks. Each intelligent filter comprises a SOM (Self-Organizing Map) component trained with local traffic and parameters/policies defined by a centralized controller. This solution suffers from a lack of validation on the trained SOM filter. For instance, if the localized traffic comprises more malicious traffic than non-malicious traffic at the training time, then the training on such data could result in incorrect detection results. Further, the real-time traffic features, though within a range often, are spread all over and thus may not yield a proper winning neuron that helps detect the DDoS attack vectors. While anomaly detection approaches trained in unsupervised/supervised mode have the potential to detect malicious traffic, relying on the same to drop suspicious traffic at the source can lead to denial of service to the customers' request at the source itself.

Our solution differs from the above works in that it offers to adapt to evolving attack vectors by giving ISPs control over incorporating the learned model and reducing the unwanted ISP resource consumption.

3. System and models

This section presents the system and models that support SmartDefense solutions, including the edge network infrastructure, communication model, and attack model. Traditional statistical approaches such as rate-limiting can block malicious traffic in a distributed network. SDN-based distributed anomaly detection system discussed in [14,15] allows detection of malicious traffic, and SDN-controller can be employed to rate-limit the malicious traffic. Another factor that needs to be considered when using source devices to defend against these DDoS attacks is the resource constraints these devices have, thus demanding a lightweight yet faster defense component. Machine learning models

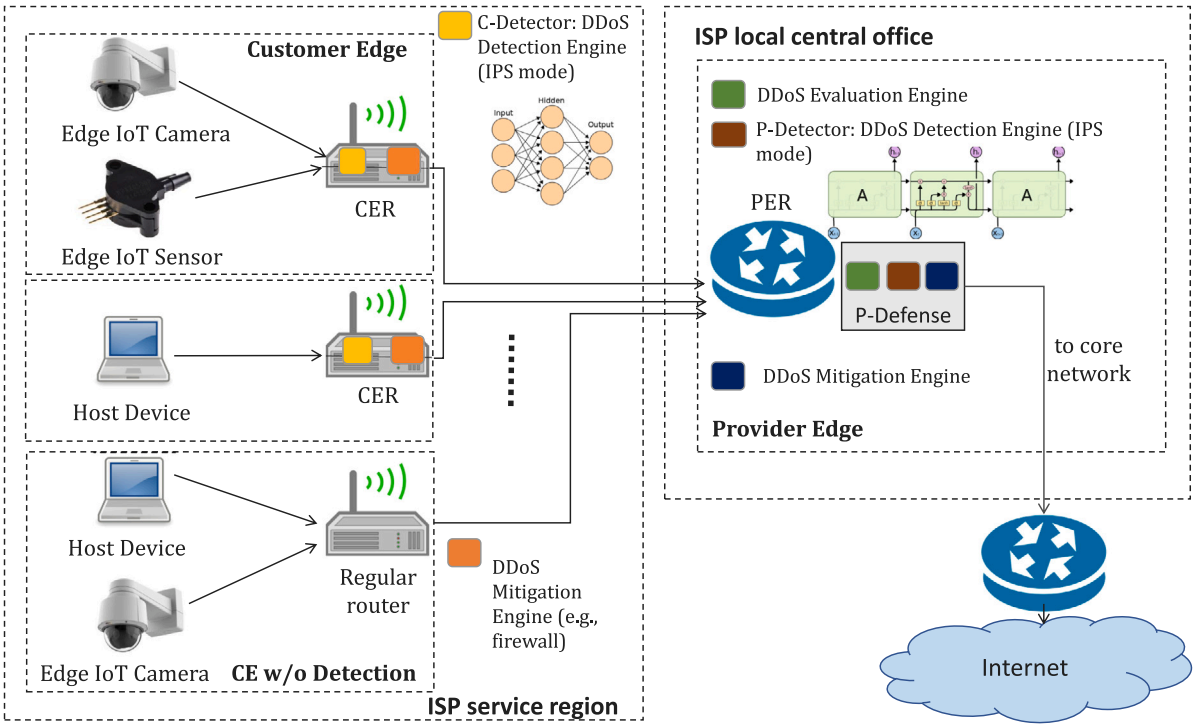

Fig. 1. CE with SmartDefense enabled CERs and regular CER. CERs monitor the outgoing traffic for any DDoS attack variants. PERs monitor the incoming traffic at PE for any DDoS attack variants.

built with deep neural networks can meet these resource constraints and huge data requirements much better than the traditional machine learning models. However, with proper utilization of edge resources, deep learning techniques can support the effective mitigation of DDoS attacks with minimal impact on legitimate requests.

3.1. System architecture

We present an abstraction of SmartDefense edge networking infrastructure Fig. 1. As part of this, the edge includes the Customer Edge (CE) and the Provider Edge (PE). The presented SmartDefense solution is a two-state DDoS detection and mitigation procedure running on CE and PE. The main components of CE and PE are programmable devices such as *Customer Edge Router* (CER) and *Provider Edge Router* (PER), respectively. A CER can be an upgraded version of a cable or a DSL modem, which runs a C-Defense component. A PER much more powerful than CER runs in PE and connects to the CER in the customer edge to route the traffic from CE. On the CE side, a lightweight DDoS detection is performed to pre-screening malicious traffic that helps reduce the workload on the PE's detection module and thus improve both efficiency and effectiveness to detect and mitigate DDoS attacks.

3.1.1. Customer edge (CE)

At each CE is a programmable device, Customer Edge Router (CER), an upgraded version of a cable modem, or a DSL modem. The main components of CE and PE are programmable devices such as *Customer Edge Router* (CER) and *Provider Edge Router* (PER), respectively. A CER can be an upgraded version of a cable or a DSL modem, which runs a C-Defense component. The C-Defense component running on the CER comprises of two modules, (a) *C-Detector*: This module runs Machine Learning (ML)-based DDoS detection models where the outcome is a probability of the outgoing traffic to be DDoS activity. This probability is passed to the second module (b) *DDoS mitigation engine*: Here, the threshold T_e, set by the ISP, will determine whether traffic will be sent to upper layers or dropped at the source. A threshold of '1' will ensure that only traffic probability of '1' to be DDoS activity will be dropped. This threshold configuration at CE can be defined differently for different customer edge networks based on the malicious traffic that the ISP has observed as originating from a given CE.

3.1.2. Provider edge (PE)

The P-Defense component running on the PER comprises of three major components, i.e., (a) an *evaluation module*, which utilizes a DDoS detection engine to perform a granular inspection of the traffic. At PE, there are two thresholds configured by ISP, (a) Threshold on traffic coming into ISP from customer edge, T_c^e, and (b) Threshold on traffic outgoing from ISP, T_c^c. T_c^e helps determine if the incoming traffic needs to be analyzed collectively with other incoming traffic. Incoming traffic's probability falling below this threshold will be passed to the second module of PE, which is (b) *P-Detector*, an anomaly detection engine that correlates network traffic from multiple sources to perform DDoS detection (c) *mitigation engine*, that utilizes the input from P-Detector to perform threat mitigation.

3.2. Communication model

The CER and PER utilize infrastructure provided by the backbone network of the content service provider (CSP) to exchange network traffic. The detection layer at the customer edge presented in Fig. 1 checks if the packets are suspicious and tag the packet flows with information that can be used for the identification of DDoS attacks. On the Smart PE, a program *Botnet Tracker* polls the mitigation engine to check if the suspicious traffic from the detection engine is marked as DDoS. The botnet tracker updates the *Botnet Engine* with a list of suspicious packets. The following information associated with each traffic flow is passing from CE to PE:

- NodeID (7 bits): This identifies a unique device at CE. With the IP address, $\langle IP, NodeID \rangle$ can track the traffic from a malicious edge device. The device traffic with reasonable suspicion based on traffic analysis at CE is tagged as malicious.

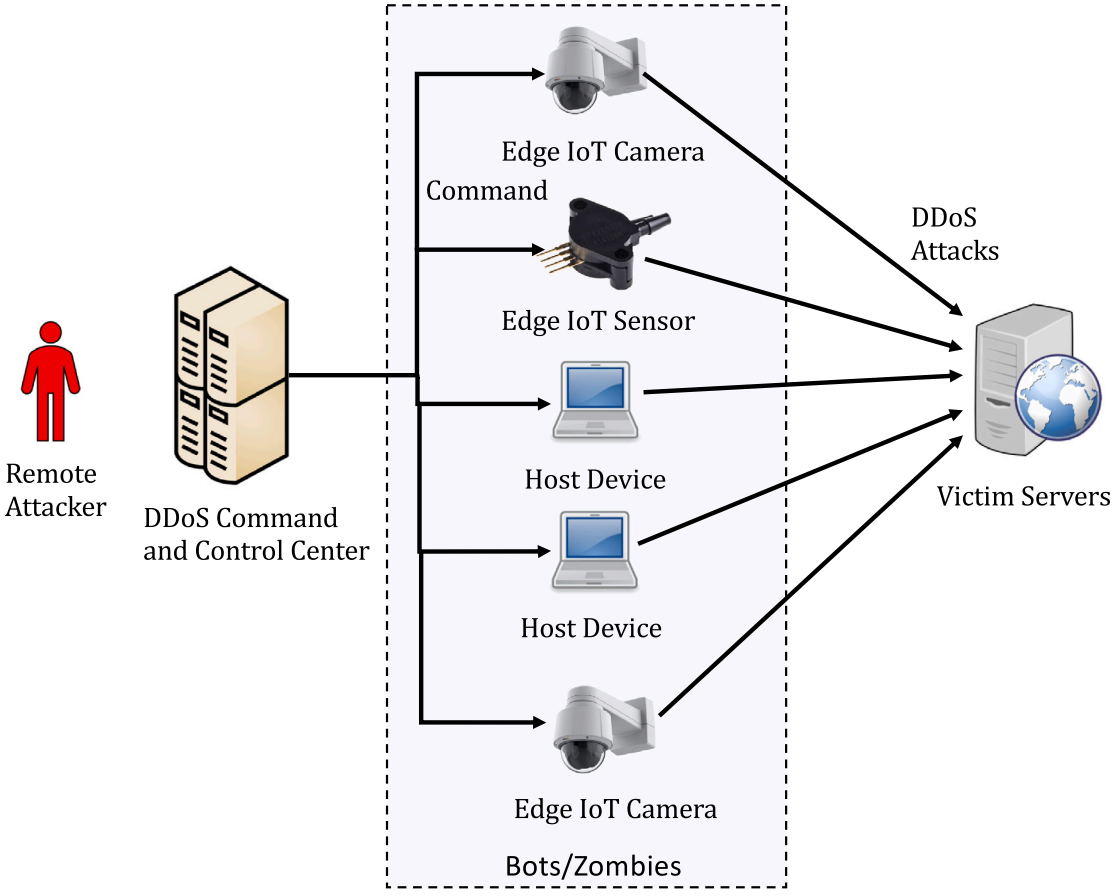

Fig. 2. A Distributed Denial of Service (DDoS) attack scenario. The attacker can use the compromised edge IoT devices such as bots/zombies to target the victim Server.

- P_p^e (9 bits): The prediction probability of the outgoing flow to be a DDoS attack flow at CE as determined by the detection engine within the C-Defense component.

The overall length of *NodeID* and P_p^e is 16 bits in length for a given customer network. We utilize opportunistic piggybacking [34] between CE and PE for tracing the packet. The traceback mechanism is initiated only when there is any suspicious activity, and routers located at the CE initiate the piggybacking. The piggybacking scheme is implemented based on a group of packets matching suspicious behavior, i.e., flow-based piggybacking instead of per-packet marking, which can increase the network latency.

3.3. Attack model

The resource constraints of IoT devices, such as the limited computational abilities, leave no space for implementation of strong security mechanisms and protocols, making these devices vulnerable [35,36]. Such vulnerable IoT devices have been found to be used as bots by attackers to perform DDoS attacks. In this work, we assume that CER and PER are trusted devices and well protected. CER has the basic security packet filtering features to block IP spoofing attacks, where packets are originated from the customer network with unknown source IP addresses. Further, attacks like TCP and UDP floods by bots in the CE network will be detected by the PER's P-Defense at the ISP edge network and acted upon by ISPs. Other known DDoS variants will be mitigated by and at the CER with its C-Defense module, thus protecting itself. PERs operating at ISP's edge network are much more powerful than CERs, and with packet filtering features, SmartCEs in customer edge, P-Defense including botnet engine are capable of protecting themselves from being a victim of an attack. We consider an attack scenario where the remote attacker attacks the victim located on the public internet using compromised zombie edge devices, e.g., IoT cameras, personal computers, IoT sensors as depicted in Fig. 2. Since the zombie devices are part of the attacker's botnet, managed by the DDoS command and control center, the attacker can induce the zombie devices to send a large volume of requests.

To have comprehensive coverage of different kinds of DDoS attacks, we analyzed DDoS attacks prevalent in real-world scenarios, emphasizing use cases focused on edge-device-based DDoS attacks. For this, we in this paper have used the CICDDoS2019 dataset published in 2019 as it has captured the most-prevalent DDoS attack variants. Table 2 shows the different types of DDoS attacks that are part of our threat model from [37]. The training set captured 12 DDoS attacks comprising DNS, LDAP, MSSQL, NetBIOS, NTP, SNMP, SSDP, SYN, TFTP, UDP, UDP-Lag, and WebDDoS. The testing set captured six attacks of the training

Table 2
DDoS attack vectors.

Attack category	Protocol	Attack vector
Reflection attack	TCP	MSSQL SSDP
	TCP/UDP	DNS LDAP NetBIOS SNMP Portmap
	UDP	NTP
Exploitation attack	TCP	SYN Flood
	UDP	UDP Flood UDP-Lag

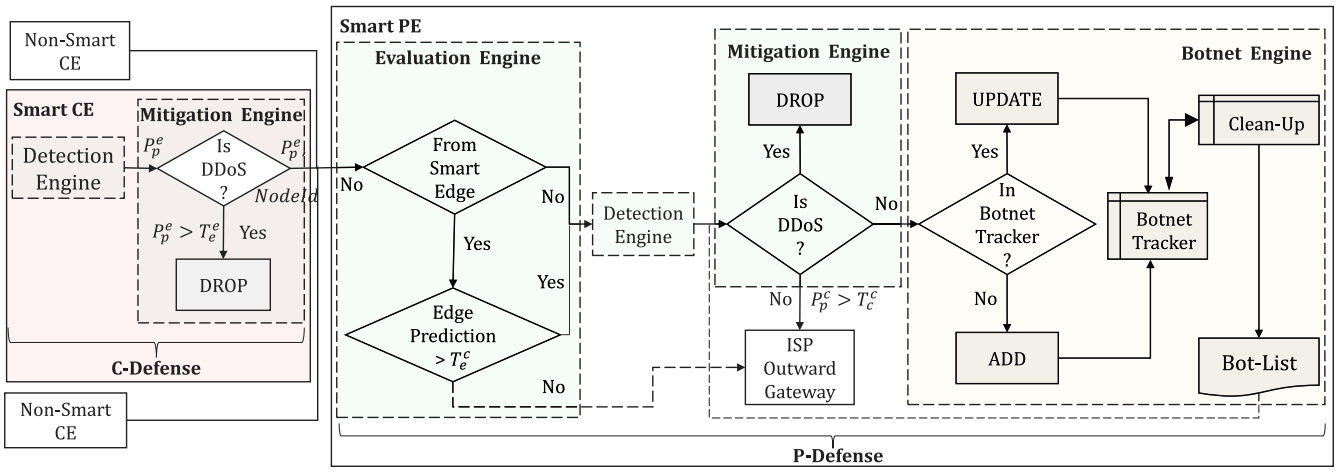

Fig. 3. Smart CE represents a residential home equipped with CER that implements SmartDefense. Smart PE, PE equipped with SmartDefense, processes traffic flows coming from both Smart CEs and Non-Smart CEs. The evaluation engine uses the attributes passed by CE to determine the need for further processing of the incoming flow within PE.

set, LDAP, MSSQL, NetBIOS, SYN, UDP, UDP-Lag, along with one new attack not in the training set, Portscan. The dataset contains 85 flow features and a Label column extracted using CICFlowMeter-V3 Lashkari et al. [38]. In Section 5.2 we discussed the details of the attack vectors used for training and testing.

4. Description of SmartDefense

4.1. C-defense function

As can be seen from Fig. 3, C-Defense is part of the CER's functions, which primarily comprises of two sub-components - (1) C-Detector: a detection engine that checks on the outgoing traffic to ensure it is not a DDoS attack variant. (2) A mitigation engine that decides to either forward or drop the traffic based on the detection engine's output.

Detection Engine: Outgoing traffic to the Internet is passed through an FNN towards detecting DDoS attacks. The FNN, having trained on a labeled dataset of multiple DDoS attack variants, looks at the incoming traffic and predicts the probability of attack traffic as specified by lines 4 and 5 of Algorithm 1. Line 4 of the procedure `AT-THE-SOURCE(R)` extracts the flow features, e.g., source IP address, destination IP address, protocol, packet inter-arrival time, etc, and makes call to `DETECTION-ENGINE(F)` to obtain prediction probability of traffic being part of DDoS attack P_p^e in line 5. Based on this prediction probability, P_p^e, the traffic is either forwarded to or dropped as determined by the mitigation engine as implemented in lines 6–10 of Algorithm 1.

Mitigation Engine: Within the mitigation engine, the input prediction probability P_p^e is checked against the threshold pre-defined by the ISP, T_e^e, to determine whether to drop or forward the incoming traffic (line 6 of Algorithm 1). This threshold initially defined by ISP can be allowed for modification by the residential homeowner. If the prediction probability P_p^e is greater than the threshold, T_e^e, the outgoing traffic will be dropped; otherwise, the traffic is forwarded along with two attributes - $nodeId$ and P_p^e. $nodeId$ is an identifier the ISP uses to identify a device within its customer's network uniquely. This $nodeId$ facilitates ISP to identify any potential bots in their customer's network, as explained in detail later in this section. P_p^e is the prediction probability of the FNN at CE used by the ISP in its CER to make decisions in an ongoing suspicious DDoS attack.

The complexity of C-Defense implementation - Algorithm 1 can be analytically described in terms of the number of traffic features, which is constant $|F|$, and number of forwarding traffic tuples $|T| = \sum(R, NodeId, P_p^e)$, that pass the FNN prediction probability threshold P_p^e, i.e, $|F| \times |T| \sim O(T)$.

Algorithm 1 C-Defense Implementation Algorithm

```
1: Input: Outgoing Traffic(R)
2: Output: Action To Take(A)
3: procedure AT-THE-SOURCE(R)
4:     F ← Extract-Features(R)
5:     P_p^e ← DETECTION-ENGINE(Model, F)
6:     if MITIGATION-ENGINE(P_p^e, T_e^e) is FORWARD then
7:         FORWARD tuple(R, NodeId, P_p^e)
8:     else
9:         DROP R
10:    end if
11: end procedure
```

4.2. P-defense function

Unlike C-Defense, P-Defense needs to operate on huge volumes of traffic that arrive at the ISP network. As shown in Fig. 3, a P-Defense comprises of three sub-components - (1) an evaluation engine that checks on the incoming traffic for specific criteria that determine the need for further detection, (2) a detection engine that checks on traffic passed by the evaluation engine towards detecting the presence of DDoS attack variants, and (3) a mitigation engine that takes action based on the detection engine's output. Additionally, a botnet engine is considered a sub-component of the mitigation engine. It maintains the traffic information forwarded to the ISP's outward gateways towards identifying suspicious bot devices in the ISP's CE. Algorithm 2 gives details of the PER implementation.

Evaluation Engine: Outgoing traffic from the edge networks, comprising of both CER and regular edge router traffic, when received by the ISP, is checked for the presence of the attributes passed by a CER. While all traffic from non-smart CE networks continues through the detection engine, only traffic from a CER carrying a Prediction Probability, P_p^e, greater than T_e^c, a pre-defined threshold by ISP, continues through the detection engine. All other CER traffic is forwarded to the output gateways of the ISP towards their destinations while passing the information on the traffic to the botnet engine for monitoring purposes. The use of CER's prediction by the PE network is the crux of our solution, $SmartDefense$. P_p^e, the prediction probability attribute from a CER together with the pre-defined threshold, T_e^c, gives ISP an ability to control the amount of traffic that continues through the detection module, specifically in case of heavy loads seen during an ongoing DDoS attack.

Detection Engine: Within the detection engine, traffic coming from the evaluation engine is passed through Long-Short Term Memory

(LSTM) based neural network. The LSTM module, having trained on a labeled dataset of multiple DDoS attack variants, looks at the current incoming traffic for the previous n (defined during training) traffic flows and predicts the probability of the current attack traffic. This prediction probability at PE, P_p^c, is then passed on to the mitigation engine to determine the action to be taken on the current traffic.

Mitigation Engine: Within the mitigation engine, the input prediction probability P_p^c is checked against the threshold pre-defined by the ISP, T_c^c, to determine whether to drop or forward the incoming traffic. If the prediction probability P_p^c is greater than the threshold, T_c^c, the outgoing traffic will be dropped; otherwise, traffic is forwarded to the ISP's outward gateways towards the destination along with sending the details of the traffic to the botnet engine. The botnet engine receives traffic originating from CER only. In the event of an ongoing DDoS attack, the mitigation engine refers to the suspicious botnet list maintained by the botnet engine to make decisions on dropping suspicious traffic that does not hit the threshold but close. This BotList holds the $NodeId$ information of the edge networks that are potentially part of botnets. In the event of a DDoS attack, this bot list serves as a ranking of the incoming traffic that bypassed the detection engine yet could be potentially part of the DDoS attack due to the traffic originating from a bot device.

Algorithm 2 P-Defense Implementation Algorithm

```
1: Input: Outgoing Traffic (R), Output: Action (A)
2: procedure NEAR-THE-SOURCE(R)
3:     if EVALUATION-ENGINE (R) ∈ {PROCESS}  then
4:         F ← Extract-Features(R)
5:         P_p^c ← DETECTION-ENGINE(Model, F)
6:         if R.NodeId in BotList then
7:             T_c^c = T_b
8:         else
9:             T_c^c ← T_o
10:        end if
11:        if MITIGATION-ENGINE(P_p^c, T_c^c) is FORWARD  then
12:            FORWARD
13:            BOTNET-ENGINE(R.NodeId, R.DestIp, R.SrcPrt, R.DestIp, R.DestPrt, P_p^c)
14:        else
15:            DROP
16:        end if
17:    end if
18: end procedure
19: procedure BOTNET-ENGINE(NodeId, SrcPrt, DestIP, DestPrt, P_p^c)
20:    TrackerTuple ← {NodeId, SrcPrt, DestIP, DestPrt, P_p^c}
21:    if TrackerTuple ∈ BotnetTrackerTable then
22:        BotnetTrackerTableEntry.Update(PacketCount)
23:    else
24:        BotnetTrackerTable.Add(TrackerTuple, PacketCount=1, Timestamp)
25:    end if
26: end procedure
27: procedure CLEANUP-BOTNET-TRACKER-TABLE
28:    for Entry ∈ BotnetTrackerTable do
29:        if Entry.SessionDuration > MinimumSessionTime & FwdPckts < MinimumFwd-
    Pckts & BwdPckts <= FwdPckts/2 then
30:            if Entry.nodeId ∈ Suspicious-List then
31:                Suspicious-List(Entry.nodeId) +=1
32:            else
33:                ADD (Entry.nodeId, 1) to Suspicious-List
34:                REMOVE Entry
35:            end if
36:        else
37:            REMOVE ENTRY
38:        end if
39:    end for
40: end procedure
```

Botnet Engine: The botnet engine within the ISP's PE network runs in the background, monitoring the ongoing traffic. The engine maintains a firewall traffic table, where it keeps track of the $NodeId$, $DestinationIp$, $SourcePort$, $DestinationIp$, $DestinationPort$, the number of the forward packets $FwdPckts$, the number of backward packets $BwdPckts$ and the prediction probability of the LSTM module P_p^c as implemented by line 13 of Algorithm 2. Each entry in the table is maintained for some time t, after which the entry will be removed as enumerated by lines 29–34. When a new packet comes in matching the entry, the number of packets in the matching entry is incremented by '1' as enumerated by lines 22–28 of Algorithm 2. During the removal of entry after time t from its creation, the total number of forwarding packets, $FwdPckts$ for the traffic flow, are checked against the threshold $MinimumFwdPckts$ and also the number of backward packets $BwdPckts$ towards determining its suspicious behavior (line 29). If $FwdPckts$ is less than $MinimumFwdPckts$, the $NodeId$ will be added to the suspicious bot list where it keeps track of the counter of its appearance on the list. Lines 24–37 of Algorithm 2 give implementation details of the creation process of the $SuspiciousList$.

This approach to detecting bots is based on a typical bot device behavior where the request is sent, but the source device does not bother to continue with that request. As a result, the number of packets for the session ends up being less than the threshold. This typical behavior referred to as *high bounce rate* and a *low session period* can be captured at PE only with the help of edge computing. A NodeId appearing more than x number of times in the suspicious list, x determined by ISP, is moved to BotList. This BotList then further is used with mitigation engine to drop DDoS traffic during an ongoing DDoS attack, where the LSTM module's prediction probability is checked against the BotList threshold, ($T_c^c = T_b$) as opposed to the original threshold ($T_c^c = T_o$) as enumerated by lines 6–10 in Algorithm 2. The algorithmic complexity for P-Defense implementation is polynomial in number of traffic flows passed to Botnet Engine, $|f| = \sum(NodeId, SrcPrt, DestIP, DestPrt, P_p^c)$, and packets from the flows that match BotnetTrackerTable, $|p|$, i.e., O(fp).

4.3. Distributed DDoS defense

The distributed DDoS defense infrastructure of the neural network is the crux of our solution, $SmartDefense$. While one might question the need for a second layer of detection when the edge has already mitigated what it knows, this second layer of detection is imperative for the following reasons:

- *Heterogeneous CEs:* For one, the traffic coming to the ISP's PE comprises traffic from both CERs and regular edge routers, making it necessary for the regular edge router's traffic at PE to go through the detection engine towards removing any known DDoS attack variants.
- *Spatial–Temporal Distribution of Traffic:* At the ISP's PE, the accumulating spatially distributed traffic exhibits collective temporal relations, the characteristic of a DDoS attack. And CER component's visibility is restricted to only the traffic from its network and thus cannot capture any temporal relations at CE. Thus, neural network models such as LSTM that can capture the time dependencies among traffic flows can detect these temporal behaviors exhibited by the incoming traffic. In Section 5 we present observations concerning spatial–temporal distribution property of the traffic.
- *Consistency:* With C-Defense, ISPs can distribute the validation load to customer edge networks that otherwise would have overwhelmed their centralized resources during peak hours, leading to inconsistencies in their service. But with no second layer of defense, ISPs have to use higher threshold values which could make slight variants of known DDoS attack patterns to bypass the C-Defense. But with P-Defense, these variants can still be caught under the context of malicious traffic accumulating at the ISP's edge network. Further, by lowering the adjustable threshold values during peak hours, ISPs will be able to ensure consistency.

Similarly, the need for CER when DDoS mitigation can happen at PER, as we show with our experimental results in Section 5 is to facilitate ISP to mitigate a DDoS attack in its early stages effectively.

- *At-the-source Mitigation:* Without CER comes the unnecessary bandwidth utilization from CE to PE due to lack of at-the-source mitigation of the DDoS traffic. While this traffic might be perceived as negligible in comparison to the huge volumes of traffic at the PE, considering the recent attack sizes in Tbps, even a small fraction of it does make a difference in reducing the overall load at the ISP, specifically during an ongoing DDoS attack.
- *Identifying Bots:* Knowing a bot in their customer's network, ISP can not only contribute to mitigating the DDoS attack traffic that otherwise reaches the cloud overwhelming the destination, but also benefit commercially. For instance, consistency is very crucial for ISPs. The information on identified bots can be used by ISPs to drop highly-probable malicious traffic coming from those bots rather than legitimate traffic during an ongoing attack. Further, during peak hours, ISPs can lower the threshold, T_e^c, to avoid further processing through the P-Defense thus ensuring consistency.

This hierarchical relation of the CE and PE networks is essential to defending against DDoS attacks. In Section 5, we present evidence emphasizing the need for a hierarchical implementation.

5. Performance evaluation

5.1. System setup

To evaluate the presented solution, we emulated a hierarchical setup of ISP Edge's PE receiving traffic from a residential network comprising ten residential homes (or CEs) in two modes - Mode-A and Mode-B. Mode-A represents a realistic scenario where some of the CEs have bot traffic, and Mode-B simulates a scenario where every CE has multiple bot devices. Mode-B differs from Mode-A in terms of network throughput. The traffic originating at each CE in Mode-B is very high than in Mode-A. Thus, Mode-A can represent a residential home, whereas Mode-B can be seen as representing IoT Gateways where each CER is run on an IoT Gateway. Further, in Mode-B, the attack data has been expanded to include the UDP-Lag attack type, which is the characteristics of some network attacks.

In Mode-A, each edge node emulates 5–15 Internet-connected devices in our system setup. For this, we have partitioned the data in the CICDDoS2019 dataset [37] by creating an IP address mapping between each CE and a subset of IP addresses from each of the attack files in the dataset, LDAP, MSSQL, NetBIOS, Syn, Portmap, and UDP activity. These attack files of the dataset comprise both normal and attack traffic. However, traffic from some IPs was only attacking traffic, and thus. In contrast, the extracted traffic includes both normal and attack traffic for some CEs. Some only network nodes have normal traffic, thus simulating a realistic scenario.

In Mode-B, both the normal traffic and the attack traffic have been distributed from each of the attack files in the test dataset, LDAP, MSSQL, NetBIOS, Syn, Portmap, UDP, UDPLag across multiple devices in 10 edge homes. Thus in this mode, every CE will have DDoS traffic. Both PE and CE in both modes were setup with Tensorflow Backend [39], NVIDIA CUDA-10.1, CuDNN-10.1 NCCL, GeoForce GT 720, and Keras library packages [40]. In our experimental evaluation, we denote each edge home using labels (E-{Number}), i.e., Edge Home 1, Edge Home 2 are denoted by E1, E2, etc.

5.2. DDoS data pre-processing

To make the CICDDoS2019 dataset Sharafaldin et al. [37] suitable for training our models, we performed the following pre-processing steps.

- *Feature Selection:* We removed the *Timestamp* and *SimilarHttp* column that we found were producing noise and not contributing to the detection of attack variant.
- *Data Cleanup:* The original data contained numerous NaN and Infinity values whose occurrence we replaced with 0 and −1 respectively in each of the dataset files, both from training and testing sets.
- *Data Normalization:* The feature data that was used to train FNN was standardized by removing the mean and scaling to unit variance. The feature data used in training LSTM was transformed by scaling each feature to range between 0 and 1.
- *Label Encoding:* We trained selected neural network models for probabilistic classification towards classifying incoming traffic flows into a range [0,1].
- *Model Architecture:* We did experiment with different combinations of hyper-parameters, including the number of hidden layers (5, 6, 7) and nodes per layer, batch size (5, 16, 32), optimizers, epochs (50, 60, 100), and also timesteps (3, 5, 10, 30). Still, the following gave us better results than we present in the latter part of this section. The FNN model was built with 5 Dense layers, 30 neurons in each layer. The LSTM model was built with 5 LSTM Layers, each with 30 neurons, taking input of 5 timesteps. A dropout of 0.2 was added on the input and hidden layers in both models as a regularization approach. While the FNN model was optimized using *adam* optimizer, the LSTM model was optimized using *relu* optimizer. In both cases, the output layer was a Dense layer with *sigmoid* activation function, and the models were trained with *binary cross entropy* loss function on the output layer for 50 epochs with a batch size of 32.

We trained both our models independently but with the same training dataset. We used the testing dataset to simulate the different scenarios, as explained in the following part of the section.

5.3. Selection of neural networks

In this section, our performance evaluation target is to show the performance gain using the presented hierarchical two-stage DDoS detection approach over the given edge networking system using different neural network models. Many neural network models have been studied to show their performance in detecting DDoS attacks in the literature. In this research, we consider two well-known solutions: Feedforward Neural Network (FNN) [41] and Long Short-Term Memory (LSTM) [42] for our performance case studies.

We must note that, in this paper, we do not try to re-prove that FNN and LSTM are capable of detecting DDoS attacks, rather point out how ISPs can mitigate DDoS attacks with these neural networks and a hierarchical defense utilizing edge computing. As such our experimental setup reflects the setup of the hierarchical networks, while also providing means to measure our solution in terms of our contributions.

Furthermore, the technical reason for choosing FNN and LSTM is based on the following facts: First, the FNN is the simplest ANN model wherein connections between the nodes do not form a cycle. The information moves in only one direction, forward, from the input nodes, through the hidden nodes, and to the output nodes. There are no cycles or loops in the network. Due to its simplicity, FNN is lightweight and can be implemented on the CE side. Second, LSTM is a Recurrent Neural Network (RNN), an offspring of FNN. It is well-suited to classifying, processing, and making predictions based on time series data since there can be lags of unknown duration between DDoS events in a time series. This feature can be used to correlate the DDoS attacks that originated from multiple attack sources.

5.4. Baseline establishment

Table 3 shows the ability of trained FNN and LSTM to detect DDoS attacks for each of the attack types along with the percentage of legitimate traffic dropped. In all the cases, LSTM has out-performed the simple FNN trained on the same data. While in the case of NetBIOS,

Table 3
Baseline establishment for FNN and LSTM.

Label	% Successfully detected		% Failed to detect		% of legitimate traffic dropped	
	FNN	LSTM	FNN	LSTM	FNN	LSTM
LDAP	90.87	100.00	9.1338	0.00336	1.50273	1.38564
MySQL	99.98	100.00	0.0191	00.00310	2.46958	1.57480
NetBIOS	99.98	99.89	0.0186	0.11362	4.54201	0.52990
PortMap	99.90	99.96	0.0963	0.04119	1.16181	0.42248
SYN	94.68	99.98	5.3191	0.01839	18.59458	1.25472
UDP	99.97	100.00	0.0325	0.0005	1.91449	4.62144
UDP Lag	61.03	98.81	38.9665	1.1864	1.14869	1.72075

MSSQL, and PortMap attacks, the FNN provided competitive performance compared to LSTM. A clear difference was observed in the case of LDAP, SYN, and UDP-Lag attacks. If we consider the UDP-Lag attack (last row from Table 3), LSTM was able to achieve an average of over 98% DDoS detection rate compared to an attack detection rate of 61% for FNN. The table further shows the percentage of legitimate traffic dropped by the trained FNN and LSTM modules at the threshold of 0.99. It is noteworthy that while LSTM has *lower* False Positive rate, and FNN had *high* False positives specifically on LDAP, Syn, UDP-Lag, the same ones where it failed to detect DDoS traffic accurately pointing to the need for further investigating the effect of additional data for these attacks in the case of FNN. While FNN had a higher false-negative rate on UDP-Lag than on Syn attack, the high legitimate drop rate in the case of SYN attack points to the FNN's lack of ability to detect flood attacks that are temporally distributed. This allows us to conclude that simple neural networks like FNN are not sufficient for DDoS attacks with complex data distribution, and time-series models like LSTM can complement FNN to detect these attacks. These three statistics present the baseline performance of trained FNN and LSTM concerning the various attacks. These are the fundamental units of the remaining experiments we present in this section.

5.5. Case 1: Single-level defense

It is often the case that both consumers and providers are not well equipped to support a system like SmartDefense. We consider the scenario where the ISP implements SmartDefense in isolation at CE or PE rather than the two-stage edge computing implementation. These experiments were conducted with the Mode-A setup of the system. The goal is to showcase that even partial implementation of our framework can help detect and mitigate DDoS attacks.

- **Defense with P-Defense Only**
  In this experiment, we have considered the case of having a SmartDefense component only at PE, in Mode-A, answering the question *Why do we need C-Defense when we have P-Defense*. Fig. 4 shows the percentage of DDoS traffic that was detected and missed at PE, respectively, on the traffic coming from the ten customer networks simulated. While it can be seen that edge homes 4, 5, 6, 7, 8, 9, and 10 do not have any DDoS traffic, DDoS traffic originating from edge homes 1, 2, and 3 were blocked at PE. Here in this experiment, PE implemented with the LSTM module was able to detect and mitigate at least 96% of the DDoS traffic accumulating at PE from these homes while only failing to detect less than 4% of the DDoS traffic. An important observation that can be made from Fig. 4 is, if the ISP has implemented C-Defense at these edge homes, it could have mitigated at least 92% of the DDoS traffic at the edge homes (lower bound for this experiment in case of edge home 3 (E3)) without having to provide the unnecessary bandwidth demanded by the DDoS traffic that ended up being dropped anyway at PE. In some cases, the C-Defense can address more than 99% of DDoS attacks (E2).

Table 4
Comparison of all cases on detection and failure rates.

Defense position	% Failed to detect	% Successfully detected
C-Defense only	2.7286%	97.2714%
P-Defense only	0.0102%	99.9898%
Hierarchy (LSTM only at P-Defense)	0.0339%	99.9661%
Hierarchy (LSTM in both components)	0.0059%	99.9941%

- **Defense with C-Defense Only**
  In this experiment, we have considered the case of having only P-Defense, in Mode-A, answering the question *Why do we need P-Defense when we have C-Defense*. Fig. 4 shows the percentage of DDoS traffic that was detected and missed at the 10 CE homes. Here in this experiment, CER implemented with the FNN module was able to detect and mitigate at least 90% of the DDoS traffic that otherwise would have accumulated at PE wasting ISP's resources. On the other hand, the C-Defense failed to mitigate 0.3% to 7% of the DDoS traffic at these edge homes, pointing out to the fact that while C-Defense can significantly contribute to mitigating the DDoS attack at the source itself, the second level of mitigation is still required at PE for effective mitigation of DDoS attacks in the very early stages.

These experiments point out that a multi-level defense mechanism is needed to defend a distributed attack like DDoS. Moreover, the SmartDefense system can be implemented at different layers in an ISP network. In the following experiment, we have implemented hierarchical defense, comprising of two-levels of mitigation, at CE, and PE.

5.6. Case 2: Two-stage edge computing defense

We consider the Mode-A setup of the two-stage edge computing implementation wherein the C-Defense was implemented with FNN and LSTM modules. In contrast, the P-Defense was implemented with LSTM. C-Defense and P-Defense actively monitor and mitigate detected DDoS attack traffic at and near the source. Table 4 presents statistics collected under emulation in Mode-A with and without the presence of C-Defense/P-Defense components along with the implemented machine learning framework. It can be observed from the table that an overall 2.7% of DDoS traffic failed to be detected at CE. However, with a hierarchical implementation, the overall detection rate improved, as can be observed from the fourth row in the table. The successful detection rate was 99.9941% when the hierarchical defense was used.

Table 5 presents the different percentages captured with the implementation of FNN in C-Defense, and LSTM in P-Defense components at varying thresholds, T_e^c within the evaluation engine at P-Defense. As the threshold was increased, the volume of traffic passed through by the evaluation engine to the detection engine at PE has reduced. This, in turn, contributed to the reduced detection rate for DDoS traffic. This accommodates that FNN at C-Defense failed to detect the SYN attack at the source, which sent the SYN attack with lower prediction probabilities. But with a lower threshold allowing all the low predicted

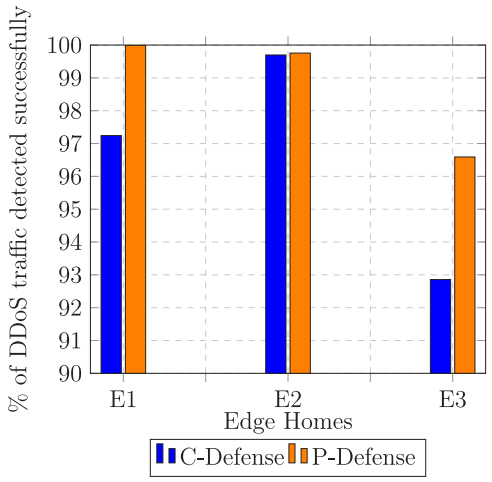

Fig. 4. Percentage of DDoS traffic detected successfully at CE vs. PE. Even, a simple FNN implementation could significantly reduce the amount of bandwidth utilized by the otherwise flowing attack traffic from CE to PE.

Table 5
Traffic detection rate based on varying traffic passed to provider edge.

Metric	Threshold = 0.0	Threshold = 0.20	Threshold = 0.50	Threshold = 0.90
%Failed to be detected	1.24%	1.5%	1.07%	0.13%
%Successfulyl detected	98.76%	84.27%	61.30%	48.20%

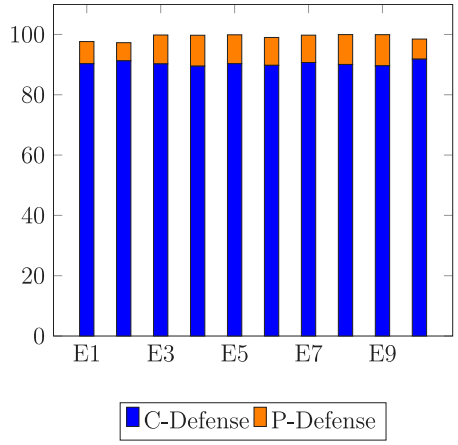

Fig. 5. Percentage of DDoS traffic detected successfully at CE and PE in Mode-B.

Table 6
Attack traffic failed to be detected by mitigation engine without and with botnet engine with $T_o = 0.99$ and $T_b = 0.90$ respectively.

Edge home#	% Failed to be detected	% Additional detection with botnet engine
E1	2.36%	46.1290%
E2	2.73%	44.9896%
E3	0.19%	0.00%
E4	0.27%	0.00%
E5	0.17%	0.00%
E6	0.51%	29.2523%
E7	0.24%	0.00%
E8	0.05%	0.00%
E9	0.08%	0.00%
E10	1.53%	51.9577%

SYN attack traffic that failed to be mitigated at source was mitigated within P-Defense, thus emphasizing the two-stage defense capabilities.

It can be observed from Table 4 that implementation of LSTM in both the SmartDefense components, C-Defense and P-Defense, results in higher accuracy and lower bandwidth utilization. However, LSTM models are computationally intensive, and deploying them at CE could overwhelm the resource-constrained CERs. Further, with the temporal associations that the edge nodes lack visibility into, implementing an LSTM within the C-Defense component will not be beneficial at the cost of intensive operations it needs. The P-Defense component capable of performing resource-intensive operations is ideal for implementing the P-Defense module with the LSTM model.

5.7. Case 3: Two-stage edge computing defense with botnet engine

Fig. 5 shows the performance of the two-stage defense under Mode-B. As can be seen, DDoS traffic, specifically, SYN attack traffic that bypassed the CEs ($T_e^c = 0.99$) due to the FNNs lack of ability to detect Syn attack as explained through Table 3, was successfully mitigated at P-Defense ($T_e^c = 0$, $T_c^c = T_o = 0.99$).

However, as shown in Table 6, there is still some attack traffic that bypassed the P-Defense component as represented by the second column statistics. We showcase that additional DDoS traffic can be mitigated with the use of the Botnet Engine. The BotList generated by the BotEngine detected bot devices using the $NodeId$ and P_p^c attributes. And at the mitigation engine within P-Defense, during an ongoing attack, with the incoming $NodeId$ in the BotList, and the threshold on the LSTM prediction probability lowered to $T_c^c = T_b = 0.90$, additional DDoS traffic was mitigated. This additional mitigation achieved was up to 51.95% of the traffic generated by the bot devices at the CE as shown by the last column in Table 6.

While in this paper, the scope of the experiments was limited to a two-stage hierarchy, the framework can support multiple layers of defenses. C-defense at customer edge and P-defense at ISP's edge is suitable for Tier-3 ISPs as they are closest to the customers. Additionally, Tier-3 ISPs can implement P-Defense in their routers at levels above their edge. For Tier-2 and Tier-1 ISPs, P-Defense without botnet engine will benefit from the accumulation of the traffic from spatially distributed DDoS sources in different regions or countries that exhibit collective temporal relations. In Tier-2 and Tier-1 ISPs, the botnet engine will be expensive to implement and not as effective as in Tier-3 due to the distance from the actual sources of the DDoS traffic.

6. Conclusion and future work

ISPs have the edge over the source and destination DDoS defense systems due to their knowledge about the ongoing traffic, the behavior patterns of the requests, and the ability to keep track of acknowledgments and responses. This paper proposes a two-stage defense system, SmartDefense, for ISPs using edge computing to address DDoS attacks. SmartDefense presents a new edge computing-based DDoS mitigation solution and utilizes a feed-forward neural network (FNN) at the consumer edge and long-short term memory (LSTM) at the ISP's edge to demonstrate the DDoS detection and mitigation capabilities of the framework. While C-Defense (with FNN) blocks easy-to-detect DDoS attacks close to the source, P-Defense (with LSTM) identifies and blocks DDoS attacks based on information correlation from multiple customer edge networks. The empirical results show how over 90% of DDoS traffic can be detected at the source, and over 97.5% of the DDoS patterns missed by FNN at customer edge can be detected by LSTM at provider edge. Thus, the overall accuracy of the DDoS detection and mitigation increases by our proposed framework, and ISPs gain the ability to detect bot devices in their customer network through the attributes forwarded by CEs. This approach, based on *high bounce rate* and a *low session period* requests coming from a typical bot device, helps ISPs to further reduce the DDoS traffic by up to 51.95% from going to the destination, as demonstrated by our experiments

While in this paper, we presented the framework and the approach in application to detecting and mitigating DDoS attacks, the framework itself is not limited to DDoS attacks. Still, it can be applied to other threats that ISPs face, such as bot devices, spamming, etc. Further, this same framework can help ISPs protect the customers from having their devices compromised with smart CERs checking the incoming traffic source against the updated blacklist provided by the ISP. Other directions of extension to this work include providing customers with anomalous behavior analysis local to the customer network. Thus, in addition to finding bot devices in one's networks, customers are incentivized by the ISP's Security-as-a-Service framework that has the potential to keep the customers' network secure. We believe this framework shows several promising research directions that can benefit the ISPs and the customers of the ISPs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Ddos attack that disrupted internet was largest of its kind in history, experts say, 2016, https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet (Accessed on 10/14/2021).

[2] February 28th ddos incident report - the github blog, 2018, https://github.blog/2018-03-01-ddos-incident-report/ (Accessed on 10/10/2021).

[3] 2019 Global ddos threat landscape report, 2020, https://www.imperva.com/blog/2019-global-ddos-threat-landscape-report/ (Accessed on 10/10/2021).

[4] Aws shield: Threat landscape report – q1 2020, 2020, https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf (Accessed on 10/14/2021).

[5] K. Bhushan, et al., Ddos attack defense framework for cloud using fog computing, in: Recent Trends in Electronics, Information & Communication Technology (Rteict), 2017 2nd Ieee International Conference on, IEEE, 2017, pp. 534–538.

[6] Q. Yan, W. Huang, X. Luo, Q. Gong, F.R. Yu, A multi-level ddos mitigation framework for the industrial internet of things, IEEE Commun. Mag. 56 (2) (2018) 30–36.

[7] 2021 Global ddos threat landscape report, 2021, https://www.imperva.com/resources/reports/Imperva_2021-DDoS-Report.pdf (Accessed on 10/14/2021).

[8] P.S. Bawa, S. Manickam, Critical review of economical denial of sustainability (edos) mitigation techniques, J. Comput. Sci. 11 (7) (2015) 855.

[9] J. Mirkovic, P. Reiher, D-ward: a source-end defense against flooding denial-of-service attacks, IEEE Trans. Dependable Secure Comput. 2 (3) (2005) 216–232.

[10] C.E. Stewart, A.M. Vasu, E. Keller, Communityguard: A crowdsourced home cyber-security system, in: Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, ACM, 2017, pp. 1–6.

[11] B.B. Gupta, M. Misra, R.C. Joshi, An ISP level solution to combat ddos attacks using combined statistical based approach, 2012, arXiv preprint arXiv:1203.2400.

[12] K. Subramanian, P. Gunasekaran, M. Selvaraj, Two layer defending mechanism against ddos attacks., Int. Arab J. Inf. Technol. (IAJIT) 12 (4) (2015).

[13] A.R. Yusof, N.I. Udzir, A. Selamat, An evaluation on knn-svm algorithm for detection and prediction of ddos attack, in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2016, pp. 95–102.

[14] F. Bannour, S. Souihi, A. Mellouk, Distributed sdn control: Survey, taxonomy, and challenges, IEEE Commun. Surv. Tutor. 20 (1) (2017) 333–354.

[15] C. Birkinshaw, E. Rouka, V.G. Vassilakis, Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks, J. Netw. Comput. Appl. 136 (2019) 71–85.

[16] I. Ko, D. Chambers, E. Barrett, Unsupervised learning with hierarchical feature selection for ddos mitigation within the isp domain, ETRI J. 41 (5) (2019) 574–584.

[17] A. Rezapour, W.-G. Tzeng, Rl-shield: mitigating target link-flooding attacks using sdn and deep reinforcement learning routing algorithm, IEEE Trans. Dependable Secure Comput. (2021).

[18] Z. Liu, X. Yin, Y. Hu, Cpss lr-ddos detection and defense in edge computing utilizing dcnn q-learning, IEEE Access 8 (2020) 42120–42130.

[19] K.J. Argyraki, D.R. Cheriton, Active internet traffic filtering: Real-time response to denial-of-service attacks., in: USENIX Annual Technical Conference, General Track, Vol. 38, 2005.

[20] S. Chen, Q. Song, Perimeter-based defense against high bandwidth ddos attacks, IEEE Trans. Parallel Distrib. Syst. (6) (2005) 526–537.

[21] Y. Chen, K. Hwang, W.-S. Ku, Collaborative detection of ddos attacks over multiple network domains, IEEE Trans. Parallel Distrib. Syst. (12) (2007) 1649–1662.

[22] Q. Niyaz, W. Sun, A.Y. Javaid, A deep learning based ddos detection system in software-defined networking (sdn), 2016, arXiv preprint arXiv:1611.07400.

[23] N.-N. Dao, D.-N. Vu, Y. Lee, M. Park, S. Cho, Maec-x: Ddos prevention leveraging multi-access edge computing, in: Information Networking (ICOIN), 2018 International Conference on, IEEE, 2018, pp. 245–248.

[24] K. Bhardwaj, J.C. Miranda, A. Gavrilovska, Towards IoT-ddos prevention using edge computing, in: {*USENIX*} Workshop on Hot Topics in Edge Computing (HotEdge 18), 2018.

[25] V.L. Nguyen, P.-C. Lin, R.-H. Hwang, Mecpass: Distributed denial of service defense architecture for mobile networks, IEEE Netw. 32 (1) (2018) 118–124.

[26] R.-H. Hwang, M.-C. Peng, C.-W. Huang, Detecting IoT malicious traffic based on autoencoder and convolutional neural network, in: 2019 IEEE Globecom Workshops (GC Wkshps), IEEE, 2019, pp. 1–6.

[27] Y. Khosroshahi, E. Ozdemir, Detection of sources being used in ddos attacks, in: 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), IEEE, 2019, pp. 163–168.

[28] Y. Li, Y. Lu, Lstm-ba: Ddos detection approach combining lstm and bayes, in: 2019 Seventh International Conference on Advanced Cloud and Big Data (CBD), IEEE, 2019, pp. 180–185.

[29] Y. Jia, F. Zhong, A. Alrawais, B. Gong, X. Cheng, Flowguard: An intelligent edge defense mechanism against IoT ddos attacks, IEEE Internet Things J. 7 (10) (2020) 9552–9562.

[30] H. Li, C. Yang, L. Wang, N. Ansari, D. Tang, X. Huang, Z. Xu, D. Hu, A cooperative defense framework against application-level ddos attacks on mobile edge computing services, IEEE Trans. Mob. Comput. (2021).

[31] D. Han, Y. Zhu, D. Li, W. Liang, A. Souri, K.-C. Li, A blockchain-based auditable access control system for private data in service-centric IoT environments, IEEE Trans. Ind. Inf. (2021).

[32] N.-N. Dao, T.V. Phan, U. Sa'ad, J. Kim, T. Bauschert, D.-T. Do, S. Cho, Securing heterogeneous iot with intelligent ddos attack behavior learning, IEEE Syst. J. (2021).

[33] S. Javanmardi, M. Shojafar, R. Mohammadi, A. Nazari, V. Persico, A. Pescapè, Fupe: A security driven task scheduling approach for sdn-based iot–fog networks, J. Inf. Secur. Appl. 60 (2021) 102853.

[34] L. Cheng, D.M. Divakaran, W.Y. Lim, V.L. Thing, Opportunistic piggyback marking for ip traceback, IEEE Trans. Inf. Forensics Secur. 11 (2) (2015) 273–288.

[35] V. Sivaraman, H.H. Gharakheili, C. Fernandes, N. Clark, T. Karliychuk, Smart iot devices in the home: Security and privacy implications, IEEE Technol. Soc. Mag. 37 (2) (2018) 71–79.

[36] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, A. Zanella, Iot: Internet of threats? A survey of practical security vulnerabilities in real iot devices, IEEE Internet Things J. 6 (5) (2019) 8182–8201.

[37] I. Sharafaldin, A.H. Lashkari, S. Hakak, 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.
[38] A.H. Lashkari, A. Seo, G.D. Gil, A. Ghorbani, Cic-ab: Online ad blocker for browsers, in: 2017 International Carnahan Conference on Security Technology (ICCST), IEEE, 2017, pp. 1–7.
[39] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {$USENIX$} Symposium on Operating Systems Design and Implementation ({$OSDI$} 16), 2016, pp. 265–283.
[40] A. Gulli, S. Pal, Deep Learning with Keras, Packt Publishing Ltd, 2017.
[41] D. Svozil, V. Kvasnicka, J. Pospichal, Introduction to multi-layer feed-forward neural networks, Chemometr. Intell. Lab. Syst. 39 (1) (1997) 43–62.
[42] K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber, Lstm: A search space odyssey, IEEE Trans. Neural Netw. Learn. Syst. 28 (10) (2016) 2222–2232.

Sowmya Myneni is a Ph.D Candidate in the Department of Computer Science at Arizona State University, Tempe, AZ, USA. She received her M.S. from Department of Computer Science at New Mexico State University in 2010. Her research interests include Network and Information Security, specifically Intrusion Detection & Prevention, Advanced Persistent Threats, Ransomware, Cryptography, Authentication and Authorization, Wireless Network Security. Besides working towards her Ph.D., she also works as a full-time Senior Security Engineer in Chandler, AZ.

Dr. Ankur Chowdhary is a cybersecurity researcher. He received Ph.D. (2020) and M.S. (2015) with specialization in cybersecurity from Arizona State University (ASU). His research interests include Cloud Security, Software Defined Networks, and application of Artificial Intelligence and Machine Learning in the field of cybersecurity. Ankur has coauthored over 25 research papers and one textbook in the field of cybersecurity. Ankur co-founded cybersecurity startup CyNET LLC (2017). Ankur has been quite active in cybersecurity education. Ankur was ASU's National Cybersecurity Defense Competition (NCCDC) captain (2015–2018), and he is current team coach (2018-). He co-founded hacking club DevilSec in 2019 to teach offensive and defensive security to students at ASU.

Dr. Dijiang Huang received his B.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, and the M.S. and Ph.D. degrees from the University of Missouri Kansas City, Kansas City, MO, USA, 1995, 2001, and 2004, respectively. He is an Associate Professor with the School of Computing Informatics and Decision System Engineering, Arizona State University, Tempe, AZ, USA. His research interests include computer networking, security, and privacy. He is an Associate Editor of the Journal of Network and System Management (JNSM) and an Editor of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS. He has served as an organizer for many international conferences and workshops. His research was supported by the NSF, ONR, ARO, NATO, and Consortium of Embedded System (CES). He was the recipient of the ONR Young Investigator Program (YIP) Award.

Dr. Adel Alshamrani has received his Ph.D. in Computer Science at Arizona State University inn 2019. He received his M.S. degree in computer science from La Trobe University, Melbourne, Australia and B.S. degree in computer science from Umm Al-Qura University, Saudi Arabia in 2010 and 2007 respectively. As a current faculty member at University of Jeddah, Jeddah, Saudi Arabia, Dr. Alshamrani leads innovative research in cybersecurity at the university. His research interests include information security, intrusion detection, and software defined networking.