Journal of Network and Computer Applications 145 (2019) 102381

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

SEAL: SDN based secure and agile framework for protecting smart city applications from DDoS attacks

Narmeen Zakaria Bawany [*], Jawwad A. Shamsi

Systems Research Laboratory, National University of Computer and Emerging Sciences, Karachi, Pakistan

A R T I C L E I N F O

Keywords:
Smart city
DDoS attacks
Software defined networking
SDN security
Smart city security

A B S T R A C T

Evolution of smart cities induces critical challenges related to cyber and network security. The increased reliance of a smart city on Information and Communication Technologies (ICT) infrastructure improves automation, efficiency, and sustainability of city services. However, it also poses enormous challenges for ensuring continued operations and services at all times and especially under cyber-attacks. Any lapse in cyber security can lead to critical disaster across the city. Distributed Denial of Service (DDoS) attacks are considered to be the most predominant and prevalent cyber-attacks. We believe that smart city could consist of numerous applications with varying level of network and security requirements. Therefore, providing an adaptive mechanism against DDoS attacks for all applications in a smart city is a key challenge. Further, considering the wide-scale requirements of a smart city, developing an adaptive and flexible solution is a key requirement. Considering these requirements, this paper presents SEAL (SEcure and AgiLe) – a novel Software Defined Networking (SDN) based adaptive framework for protecting smart city applications against DDoS attacks. The SEAL framework leverages key characteristics of SDN such as the global visibility, centralized control, and programmability to enhance the security and resilience. SEAL is capable of effectively detecting and mitigating DDoS attacks not only on application servers but also on network resources. SEAL is also unique in this regard that it provides application specific DDoS attack security solution instead of static threshold mechanism. Moreover, inherently distributed architecture of the SEAL framework ensures fault tolerance, scalability and reliability of the smart city. The SEAL framework comprises three modules, namely D-Defense, A-Defense and C-Defense. These modules collectively provide a mechanism to detect and mitigate DDoS attack on smart city applications and the network infrastructure. Adaptability in SEAL is achieved through implementing customized version of estimated-weighted moving average (EWMA) filters. Three types of filters, Proactive Filter, Active Filter, and Passive Filter are proposed and implemented to compute the dynamic threshold in real time for various types of applications. Experimental evaluation of the SEAL framework has been conducted to establish the efficacy of the framework and its components in detecting and mitigating DDoS attacks. The results prove that SEAL is able to detect and mitigate DDoS attacks effectively. The focus of the SEAL framework is to protect smart city applications, however, the SEAL framework can potentially be utilized in a wide range of systems.

1. Introduction

Ensuring cybersecurity in a smart city is of a paramount importance, as many applications are interconnected in a smart city network serving critical and vital facilities. A smart city entails complex infrastructure of data collection, integration, aggregation, information extraction, and process control. Any disruption or interruption to smart city online services due to cyber-attacks may lead to catastrophic consequences and disasters for the city's economy and citizens (Bawany and Shamsi, 2016)(C. Cerrudo, 2015). Numerous cyber-attacks on critical smart city infrastructure including smart grid, industrial control structure, etc. have been reported (Kimani et al., 2019) (Maziku et al., 2019) (Kitchin, 2016).

DDoS attacks have become one of the major cyber security threats. (Hiller and Russell, 2013). Such attacks are one of the most impactful form of cyber-attacks which have the potential of causing mass destruction on smart city applications and networks. (C (IOActive L. Cerrudo, 2015)(Alibasic et al., 2016).

Ensuring sustainable security for smart city scale is challenging due to enormous requirements of scalability, integrity, intrusion prevention, and timeliness (Sookhak et al., 2018). Further, a smart city may contain

* Corresponding author.
E-mail address: narmeen.bawany@nu.edu.pk (N.Z. Bawany).

https://doi.org/10.1016/j.jnca.2019.06.001
Received 17 November 2018; Received in revised form 11 April 2019; Accepted 1 June 2019
Available online 24 July 2019
1084-8045/© 2019 Elsevier Ltd. All rights reserved.

different applications with varying requirements of network resilience and safety. For instance, a smart power grid system(Wang and Lu, 2013) may not be able to tolerate any interruption in utility networks as this may have a severe impact on city operations. Comparatively, a smart car parking system (Kianpisheh et al., 2012) may be able to endure a network outage for a few seconds due to lighter requirements of availability. A DDoS attack may have different implications for these two applications.

In reality, smart city may contain various applications with different requirements of network availability and security. Few applications that are critical in nature will have stringent requirements of security and availability. For such applications, rapid detection of an attack is essential. Further, these applications cannot tolerate any false negative cases from the security solutions. In contrast, there are certain applications with comparatively moderate or less stringent requirements of availability. An important factor in defending against DDoS attacks in such scenarios is the correct implementation of hard and soft threshold values for each type of applications. Hence a classification mechanism is needed to categorize smart city applications according to their tolerance to DDoS attacks.

Developing a generic solution which can provide protection against DDoS attacks for all such applications is a significant requirement. Moreover, an efficient mechanism is needed which can incorporate adaptive network management policies to mitigate DDoS attacks, according to requirement of applications. Considering the mega scale requirements of a smart city, the solution should be adaptive and flexible in order to cater for instant updates and network outages.

The above mentioned requirements are the prime considerations of our work that makes it novel and unique. In this research, we are motivated by the key fact that DDoS attacks can be a major barrier to sustainable services of smart cities. Specifically, our primary goal is to provide an effective DDoS attack prevention mechanism for smart city services and applications. To this end, we propose and implement SEAL (SEcure and AgiLe) – a novel Software Defined Networking (SDN) based adaptive framework for protecting smart city applications against DDoS attacks. The use of SDN provides enhanced flexibility in controlling network policies and separating network devices and control logic.

SEAL provides application-specific DDoS attack detection and prevention mechanism. For this purpose, applications are classified into three categories with respect to the impact of DDoS attack. These are: critical; moderate; and low impact applications. This classification has been made according to the impact a DDoS attack can have on an application.

The SEAL framework incorporates an application specific security criterion to detect DDoS attacks by using adaptive threshold levels. This feature is implemented using customized version of estimated weighted moving average (EWMA) filters that predict the occurrence of an attack on the basis of real-time network traffic. Three types of filter (a) Proactive Filter, (b) Active Filter and (c) Passive Filter are proposed and implemented to satisfy the need of various categories of applications. Proactive filter is appropriate for applications that have stringent security requirements. Critical smart city applications requires high level of security hence the Proactive Filter serves as an effective defense mechanism for such systems. This filter detects rising network traffic trends and immediately put mitigation strategy in place minimizing the risk of potential attacks. Similarly, Active Filter and Passive Filter fulfills the need of applications demanding moderate to soft threshold values.

This integration of application classification and EWMA filters empowers SEAL to provide an agile and adaptive DDoS attack detection and mitigation solution with varying degree of sensitivity to possible attacks.

The SEAL framework comprises of three modules, namely, A-Defense, D-Defense and C-Defense. The A-Defense module is responsible for detecting DDoS attack on applications. The objective of the A-Defense module is to detect and mitigate an attack before the application server is completely overwhelmed with malicious packets. To accomplish this goal, A-Defense uses fast, effective and lightweight entropy based mechanism to detect an attack. The D-Defense and C-Defense modules are responsible for protecting network infrastructure, that is forwarding devices such as routers and switches (also known as data plane) and control logic of the network (i.e., control plane) from DDoS attacks. Together, the three modules are responsible for ensuring high level of security.

All three modules of the SEAL framework have been extensively evaluated through experimental analysis using a testbed. These experiments demonstrate the effectiveness of the solution in detecting and mitigating DDoS attacks on smart city applications, data plane layer and control plane layer. Additionally, resiliency of control plane layer has been thoroughly evaluated to ensure the consistent performance of controller.

The framework does not require any third party devices and all security functions are implemented and enabled within network devices. Hence, no middle-boxes are required in contrast to traditional network security solutions. The SEAL framework is capable of encountering both internal and external threats as each SDN-enabled device can be configured to detect and mitigate DDoS attacks.

Our previous work (Bawany et al., 2017), was focused on introducing a fundamental framework for DDoS attack detection through SDN. This work extends the previous work along multiple dimensions.

- We define the threat model and security requirements for smart city applications (Section 3.1)
- We propose and design an architecture of the SEAL framework, aimed at providing effective and agile DDoS attack detection and mitigation solution (Section 4)
- We propose an application specific security criterion to detect DDoS attacks by using adaptive threshold levels (Section 3).
- We design and provide implementation of D-Defense module (Section 4.1), A-Defense module (Section 4.2) and C-Defense module (Section 4.3) that is intended to detect and mitigate attacks at various levels.
- We propose and implement a load balancing algorithm to meet resiliency, scalability and performance requirements of control plane (Section 4.3.1).
- We conducted extensive evaluation of the SEAL framework to study its effectiveness in detecting DDoS attacks (Section 4.1.2, 4.2.2, 4.3.2).

The above contributions are summarized in Table 1. We anticipate that the SEAL framework would be highly beneficial in detecting and mitigating DDoS attacks in smart cities.

The remainder of the paper is organized as follows. Section 2 presents background and related work. Section 3 defines the smart city applications and related security requirements along with the threat model. Section 4 describes the SEAL framework and its components. Section 5 concludes the paper, and Section 5 highlights the future work.

2. Background and related work

DDoS attacks have become one of the major cyber security threats over the last decade for all ICT based critical infrastructure (Hiller and

Table 1
Main contributions of this research.

Contributions	Sections
Threat model for smart city applications is defined	3.1
The SEAL framework is proposed to protect smart city applications and infrastructure from DDoS attacks	4
Various modules of SEAL, D-Defense, C-Defense and A-Defense are designed and implemented to protect data plane, control plane and applications from DDoS attacks	4.1, 4.2 4.3
Load balancing algorithm to meet resiliency, scalability and performance requirements of control plane is presented	4.3.1
Experimental evaluation is conducted and detailed results are presented	4.1.2, 4.2.2, 4.3.2

Russell, 2013). Smart cities are expanding their reliance on ICT, thereby becoming a potential target for DDoS attacks. Many cases has been reported in which smart systems of countries suffered due to DDoS attack. For example, it was reported in June 2016 that many Irish government websites were not accessible due to DDoS attack. Similarly, Thai government website and Polish airlines were inaccessible due to DDoS attack in 2015 (Bawany and Shamsi, 2016). More recently, American cities also came under cyber-attack.[1]

In DDoS attacks, victim is overwhelmed with the huge amount of malicious network traffic resulting in a denial of service or performance degradation. These attacks emanate from a large number of distributed sources at once, and attackers forge the IP addresses of the sources making it more difficult to combat. Some common types of volumetric DDoS attacks are protocol exploitation flooding attacks and reflection-based flooding attacks. In protocol exploitation flooding attacks, specific features of a protocol are exploited such that resources of target server are consumed excessively. Common examples of such attacks are TCP SYN flood attack and UDP Fragmentation attack (Kaufman et al., 2003). In reflection-based flooding attacks, reflector servers are used to send attack traffic to a victim, eventually hiding the attackers' identity. The attacker initially send the requests to reflector servers using the source IP address of victim. The reflector servers send their replies to the victim, thus exhausting its resources. This kind of attack is particularly difficult to detect as reflector servers are, typically, legitimate servers. Common examples of such attacks are Fraggle Attack and Smurf Attack. In Fraggle attack, a large number of UDP packets are sent to a network using a broadcast address. The source IP address of these packets is forged with victims IP address. This generates huge amount of illegitimate traffic on the network causing network congestion. Smurf attack is similar to Fraggle attack but it uses illegitimate Internet Control Message Protocol (ICMP) traffic instead of UDP traffic to consume the victims' resources.

The SDN architecture has been leveraged by many researchers to detect and mitigate DDoS attacks and improve the network security and resiliency. The SDN refers to a revolutionary network paradigm that enables innovation in design and management of networks. The key characteristic of SDN that distinguishes it from traditional networking is the separation of control plane and data plane.

This decoupling of control and data plane helps in striping the complex control logic out of the network elements such as routers, firewalls etc. and form a logically centralized controller to regulate packet forwarding across the network. The software based flow analysis is a powerful feature of SDN that is useful in detecting DDoS attacks. Flow statistics may reveal anomalies triggered by malicious activities.

The following subsections presents the comprehensive survey of SDN based DDoS attack detection and mitigation techniques. We have divided this section into two sub-sections. The first sub-sections discusses the attack detection and mitigation on data plane followed by the existing strategies employed for protecting control plane.

2.1. Attack detection on data plane

One of the earliest work that utilizes flow based feature of SDN to detect DDoS attacks was presented by Braga et al. (2010). The DDoS attack detection method is based on collecting network traffic data periodically from a NOX (Gude et al., 2008) controller. The network flow features are analyzed using Self Organizing Maps (SOM) and then classified as normal or malicious. The network flow features that are used include pair-flows percentage, packets per flow average, bytes per flow average, duration per flow average, growth of single-flows and growth of different ports. However, the controller performance is expected to be degraded while monitoring, maintaining and processing this huge amount of data (Alsmadi and Xu, 2015), specifically, in a centralized controller environment.

Mehdi et al. (2011) assesses the implementation of various traditional anomaly detection methods on SDN. The proposed framework comprises of four anomaly detection algorithms that detects malicious traffic in home and office networks. The algorithms, which were implemented on NOX controller, includes threshold random walk with credit based rate limiting (TRW- CB algorithm), rate-limiting, maximum entropy detector and NETAD. However, the work is focused on network traffic based on home environment. Besides, mitigation strategy for the of the detected network anomalies is not discussed.

Chu et al. (Yu Hunag et al., 2010) proposes a simple method to detect a possible DDoS attack. The DDoS defender application detects the DDoS attack by monitoring the number of flows in OpenFlow switch. Once the volume of flows exceeds the predefined threshold the controller considers it a DDoS attack and inserts a flow rule to drop packets.

In another work, a fuzzy logic based system is implemented by Dotcenko et al. (Dotcenko et al., 2014) to identify hosts engaged in malicious network scanning. The combination of TRW-CB and Rate Limiting algorithm along with fuzzy logic inference is used to detect malicious traffic. The experiments are carried out using Beacon controller and Mininet on a small topology comprising of one OpenFlow switch and four hosts.

Piedrahita et al. (2015) proposed FlowFence, a congestion avoidance technique for attack mitigation. The FlowFence architecture comprises POX (Ligia Rodrigues Prete et al., 2014) controller and OpenFlow routers running applications that monitors the bandwidth consumption of interfaces. The average bandwidth usage is used to detect congestion conditions. The experiments were performed using the Future Internet Testbed with Security (FITS) on small topology having two OpenFlow routers and four virtual hosts running on two physical machines.

All of the above mentioned work uses native OpenFlow statistics collection method. This method requires access to the counter of each flow entry in the flow table. Using OpenFlow to collect statistics induces lot of processing overhead for centralized control plane in large scale networks. This in turn introduces performance and scalability issues (Giotis et al., 2014).

To overcome this problem many researchers have preferred using sFlow (Buragohain and Medhi, 2016) with OpenFlow. The sFlow collection technique separates network traffic statistics collection process from the forwarding logic that minimizes the processing overhead of the controller. Giotis et al. (2014) proposed anomaly detection technique combining OpenFlow and sFlow. The solution comprises of three modules, collector module, anomaly detection module and mitigation module, implemented as NOX applications. The collector module employs sFlow monitoring that separate the flow statistics collection process from control plane. The anomaly detection mechanism use entropy based algorithm to detect DDoS attacks and pass the related information to anomaly mitigation module that takes the appropriate action.

Another method proposed by R. Wang et al. (R. Wang et al., 2015) aims at reducing the communication overload between SDN controller and OpenFlow switches. The proposed DDoS flooding attack detection technique uses entropy based approach. The mechanism is implemented in the OpenFlow switch instead of SDN controller. The experiments were carried out using Floodlight controller. Buragohain et al. (Buragohain and Medhi, 2016) proposes a DDoS detection and mitigation for data center called FlowTrApp. The application uses sFlow to collect the network traffic statistics. The threshold on flow rate and flow duration is used for attack detection. FlowTrApp also includes attack detection and mitigation system for web based applications. The experiments were conducted on Mininet using a small topology with 20 switches and 16 hosts.

The purpose of these studies is to reduce the load on control plane by reducing the communication between switches and control plane. Though, the above mentioned work utilizes sFlow to minimize the processing overhead in large scale networks, centralized controller, used in

[1] https://diginomica.com/2018/03/30/americas-cities-cyberattack-thats-bad-news-iot-smart-cities/.

these studies, will be overwhelmed due large number of network devices connected to it. The centralized controllers such as POX, NOX, Floodlight, do not provide built-in scalability and fault tolerance mechanisms needed to meet the requirements of such networks.

2.2. Attack detection on control plane

Many researchers have addressed the security issues of flooding attack on control plane(Technol, 2015). Such attacks exploit inherent communication bottleneck that arises between the control plane and the data plane. Shin et al. (Seungwon Shin et al., 2013a,b) proposed a framework using POX controller to overcome this problem. The framework tackles the control plane saturation issue by adding intelligence to the OpenFlow data plane. However, the work is focused on detecting TCP-SYN flood and network scanning attacks and uses a centralized controller.

Phan et al. (2017) uses an artificial neural network based approach that incorporates a Distributed Self Organizing Map system within OpenFlow switches to encounter flooding attacks. The proposed system also addresses other issues of control plane, such as performance bottleneck and controller overloading. The security modules are installed in OpenFlow switches. Security application is deployed in the application layer which controls each of the modules running inside OpenFlow switches. However, the experiments were carried out on a small topology comprising of three OpenFlow switches and a centralized controller, POX.

Wang et al. (H. Wang et al., 2015) introduces FloodGuard which is an efficient and protocol-independent framework for protecting control plane. FloodGuard is specifically designed to alleviate flooding attack, which can saturate data to control plane channel. FloodGuard comprises of two modules, that is, proactive flow rule analyzer module and packet migration module. The proactive flow rule analyzer dynamically derives flow rules based on runtime logic of controller and the applications running on top of it. Packet migration buffers the flooding packets and send them to controller using rate limit and round-robin scheduling algorithms. The experiments are conducted using POX controller on a small topology.

Sahay et al. (2017) proposes a DDoS defense framework for Internet Service Provider (ISP), ArOMA that consists of traffic monitoring, anomaly detection and mitigation modules. ArOMA allows the ISP to enforce the mitigation policy according to the security alerts sent by the customer. The focus of this work is on DDoS mitigation and detection of attack is left to the customer who shares the attack alerts with its ISP via a communication channel between the SDN controllers deployed at the ISP and customer sides. The prototype is implemented using Ryu (Khondoker et al., 2015) controller connected to a fourteen switch network topology simulated on Mininet.

In order to cope with DDoS attacks on specific application server, Lim et al. (Lim et al., 2014) proposed a DDoS Blocking Application running on POX controller. The experiments were conducted using emulation on Mininet. Suh et al. (2010) propose a content-oriented networking architecture (CONA). This is an agent based architecture where agents deliver the contents requested by the user. In case of DDoS attack, the agent will first detect the storming incoming requests toward the target application server. When the server receives more requests than a predefined range, an attack detected. Evaluation of the CONA prototype is conducted on the testbed made up of two NetFPGA-OpenFlow switches and two desktop computers as hosts.

A centralized control plane in SDN has many advantages such as network visibility and centralized control, it also imposes potential scalability and reliability issues. The control plane can not only become a bottleneck in large scale network but may also turn out to be a single point of failure. To overcome these issues, researchers have proposed logically centralized but physically distributed SDN controllers. There are many distributed controller platforms such as OpenDaylight (Medved et al., 2014), ONOS (Berde et al., 2014), Onix (Koponen et al., 2010), HyperFlow (Tootoonchian and Ganjali, 2010), Kandoo (Hassas Yeganeh et al., 2012), DISCO (Phemius et al., 2014), etc, which recommends the use of multiple SDN controllers with in a control plane. Placement of multiple distributed controllers will improve the performance and scalability for larger networks.

Zhou et al. (Zhou et al., 2015) presents a load balancing approach for distributed controller architecture. The proposed dynamic and adaptive algorithm for controller load balancing, referred to as DALB, has no centralized component. The DALB runs on each Floodlight controller, hence load balancing decision can be taken locally. The experiments were conducted using two controller cluster connected with 8 and 11 switches.

Abdelaziz et al. (2017) proposed a distributed controller cluster to enhance the scalability, reliability and performance of SDN. Each cluster comprises three nodes and one controller acts as the primary controller. The proposed architecture is deployed and tested on Amazon EC2 cloud servers using HP VAN SDN controller and ONOS controller. If a primary controller fails, a backup controller from within a cluster becomes primary. This ensures the reliability of control plane. The primary controller is selected based on priority configuration. The emulation results validates that network is able to handle unexpected load. Also, continuous network operation is observed even in case of primary controller failure.

Mousavi et al. (Mousavi and St-Hilaire, 2015) proposed a solution to protect SDN controller against DDoS attacks. The author proposed entropy based solution to detect such attacks. Experiments were conducted using POX controller connected to tree type network of nine switches and sixty four hosts. Dharma et al. (2015)proposes a defense mechanism to detect attack on SDN controller. The proposed method considers destination IP addresses, time and attack pattern to detect and predict an attack. The evaluation is done using Mininet and OpenDaylight controller connected to four switch topology.

To improve the availability of control plane under DDoS attack Yan et al. (2017) proposed a time slice allocation based scheduling algorithm. The intensity of DDoS attack results in various time slice allocation strategies. All process flow requests, coming from switches under attacks and normal switches, are scheduled by the SDN controller. Experiments are conducted using Mininet and POX controller.

Han et al. (2017)presents a DDoS attack defense framework with collaborative intelligence called OverWatch in order to protect hosts and servers inside a particular network. OverWatch implements a lightweight flow monitoring algorithm that detects the DDoS attack on the basis of network traffic volume feature and asymmetry feature. Control plane utilizes a machine learning based DDoS attack classifier and a botnet tracking algorithm to determine the attack type and botnet locations. Experiments were conducted using a physical topology of eight hosts and a Ryu controller connected to FPGA-based OpenFlow switch.

In order to address the issue of unexpected network load fluctuations and the scalability in distributed controller environment, Shang et al. (2017) proposed a Service Oriented Load Balancing Mechanism. A load balancing algorithm is implemented that takes into account flow-requests, current load and propagation delay of idle controllers. The controller triggers the overload state when the current load by passes the pre-defined value and the number of flow request crosses the predefined mean value. The flow requests are then assigned to idle controllers or the controller with less mean load value. The experiments are conducted using Floodlight controllers on Mininet. Zaalouk et al. (2014) proposed orchestrator-based architecture for developing network security application by implementing SDN control functions. OrchSec provides monitoring functionalities at different levels such as varying levels based on overall network security requirements. Using OrchSec framework various applications for mitigating against network attacks such as ARP Spooing/Cache Poising, DoS/DDoS were implemented. Mininet was used for creating a testbed with a small topology having three hosts along with POX and Floodlight controllers.

Chen (Chen et al., 2016) proposed a more comprehensive defense against DDoS attacks on both OpenFlow switches and centralized controller. The proposed solution called SDNShield deploys customized software switches termed as attack mitigation units (AMU) to overcome the bottlenecks at the SDN network edges. It further integrates a two stage filtering mechanism to protect the centralized SDN controller. However, a small topology comprising of four OpenFlow switches is used for the experiments.

This research is motivated by the above mentioned work. However, the existing SDN based DDoS attack detection and mitigation solutions has few serious limitations that are addressed by SEAL. Summary of comparison is given in Table 2. The work discussed in (Braga et al., 2010) (Yu Hunag et al., 2010) (Mehdi et al., 2011) is limited to detection of DDoS attacks on data plane only. Various methods to ensure availability of control plane, in case of DDoS attack, are presented in (Zhou et al., 2015) (Mousavi and St-Hilaire, 2015) (Dharma et al., 2015). Very limited work (Suh et al., 2010) (Lim et al., 2014) was found on protecting the application servers running on SDN. Moreover, most of the experiments are conducted on centralized controllers and small topologies. In contrast, the SEAL framework is capable of detecting and mitigating DDoS attacks not only on data plane and control plane but also provides a mechanism to protect flooding attacks on application servers. Furthermore, SEAL provides an application specific security using filters well suited for smart city applications as against the static threshold mechanism in most of the existing work. The test bed was developed using a distributed controller platform, ONOS, and experiments were conducted using large topology.

Table 2
SDN based DDoS attack detection and mitigation mechanisms.

Year	Authors	Data Plane	Control Plane	Application server	Controller
2010	(Braga et al., 2010)	✓	×	×	NOX
2010	(Yu Hunag et al., 2010)	✓	×	×	NOX
2010	(Suh et al., 2010)	×	×	✓	NOX
2011	(Mehdi et al., 2011)	✓	×	×	NOX
2013	(Seungwon Shin et al., 2013a,b)	✓	×	✓	POX
2014	(Dotcenko et al., 2014)	✓	×	×	Beacon
2014	(Giotis et al., 2014)	✓	×	×	NOX
2014	(Lim et al., 2014)	×	×	✓	POX
2014	(Zhou et al., 2015)	×	✓	×	Floodlight
2015	(Piedrahita et al., 2015)	✓	×	×	POX
2015	(R. Wang et al., 2015)	✓	×	×	Floodlight
2015	(H. Wang et al., 2015)	✓	×	×	POX
2015	(Mousavi and St-Hilaire, 2015)	×	✓	×	POX
2015	(Dharma et al., 2015)	×	✓	×	Opendaylight
2016	(Buragohain and Medhi, 2016)	✓	×	×	Floodlight
2016	(Chen et al., 2016)	✓	✓	×	Centralized
2017	(Phan et al., 2017)	✓	×	×	POX.
2017	(Abdelaziz et al., 2017)	×	✓	×	ONOS
2017	(Sahay et al., 2017)	✓	×	×	Ryu
2017	(Han et al., 2017)	✓	×	×	Ryu
2018	SEAL	✓	✓	✓	ONOS

3. Smart city applications and related security requirements

Smart city applications are diverse not only in their domain but also in terms of security requirements. Few applications that are critical in nature will have stringent requirements on security and availability. For such applications, rapid detection of an attack is essential and they cannot tolerate any false negative cases from the security solutions. In contrast, there are certain applications with comparatively moderate or less stringent requirements on availability. Such smarty city applications expect flash crowd phenomenon frequently, therefore, can tolerate some delay in network availability. We presented a similar classification in our earlier study (Bawany et al., 2017).

This research focuses on development of customized DDoS attack prevention solution for smart city applications. Primarily, smart city applications are classified with respect to their cyber security requirements and their impact in case of failure. Subsequently, these applications are divided into three categories; namely: Critical; Moderate; and Low Impact, as shown in Table 3.

The applications providing crucial services, such as smart traffic control, smart grid, have stringent and unique requirements for cyber security. This calls for a highly effective and efficient attack detection mechanism with very low false negative rate. Clearly, these services cannot afford an interruption hence an extremely agile security solution is required with a proactive approach.

The detection and mitigation solutions must go an extra mile to fulfill the requirements of applications in critical category. In contrast, applications providing information services to citizens such as updates on weather, air pollution, etc. have less stringent cybersecurity requirements. Such applications entail a more pragmatic approach having low false positive rate. Therefore, a moderately agile security solution is applicable in these cases as shown in Table 3.

3.1. The threat model

Smart city is largely dependent on information and communication technologies, it brings along its inherent threats such as identify theft, data manipulation, data sovereignty, digital trails, cyber-attacks (Das et al., 2019) (Clive James, 2016). Cyber-attacks are one of the critical threats on a smart city network due to its potential to create a disastrous impact in a city. DDoS attacks had been increasing in magnitude and frequency recently making it among the top cybersecurity threat. The protection of smart city services from DDoS attack has now become a paramount priority for all smart city stake holders as it becomes a focal point for adversaries (Vitunskaite et al., 2019).

Table 3
Categories of smart city applications and related security requirements.

Application Examples	DDoS attack impact	Security requirements
Smart Traffic Control System, Smart Grid, Smart Emergency Response, Smart Health Care	Critical	Extremely Agile The security solution must take the proactive approach, as the impact of attack is disastrous. Attack detection must have low false negative rate
Location based services, Smart Agriculture	Moderate	Agile The security solution is expected to be pragmatic as the impact of attack is critical. An adequate solution that neither reacts before time nor delays the alert is required
Weather update, News, Smart Parking	Low	Moderately agile The security solution must take the reactive approach as the continuity of services is more important. Attack detection must have low false positive rate

The scope of our work is focused on volumetric DDoS attack on smart city datacenter hosting smart city services. The adversaries can launch volumetric attack on both the datacenter infrastructure and the application services to knock out the servers. We propose DDoS attack defense framework for smart city applications using SDN therefore the threat model includes both SDN infrastructure and smart city applications specific threats. The two key attack vectors are listed below:

i. Attack on infrastructure layer
ii. Attack on smart city application servers

The threat model is derived after carefully exploring the vulnerabilities plaguing the SDN based smart city datacenter.

3.1.1. *Attack on infrastructure layer*

The infrastructure layer threat model takes into account attack on two critical layers of SDN that is, data plane and control plane. Fig. 1 illustrates the attack on data plane as Threat A and the attack on control plane as Threat B.

In data plane attacks, aim of an adversary is to exhaust the network bandwidth of the network infrastructure by flooding the network with malicious packets. An adversary uses large number of compromised machines, called bots, to launch an attack. Various types of flooding attack can be launched which includes ICMP flood, UDP flood, and other spoofed packed flood. Such attacks typically generate large volumes of network traffic that can saturate the network bandwidth leading to denial of service.

In control plane attacks, an adversary attempts to overwhelm the controller by inducing large number of control messages. In contrast to traditional networks, the logically centralized control plane in SDN is responsible for making decisions such as resource request, exchange of states and network configuration. SDN control plane not only has a critical role in management of entire network but also it is the only decision making entity. Therefore, it becomes a primary target for adversaries. Under normal operations an OpenFlow switch forward the packets that matches its flow rules entries. When a new flow arrives at an OpenFlow switch packet_in message is sent to the controller. A packet_in represents a packet that does not match any flow rules in OpenFlow switch and such packets are sent directly to the controller by the switch for further action. Even if the cluster of controllers is used, there is an inherent imbalance between the number of OpenFlow switches in the data plane and the number of controllers in a control plane. There is a potential of bottleneck when OpenFlow switches send high frequency flow request transactions. An adversary can employ large number of bots to simultaneously generate forged flows which in turn generates huge amount of control messages. This will deplete SDN control plane resources resulting in a degradation of controller performance or a complete denial of service.

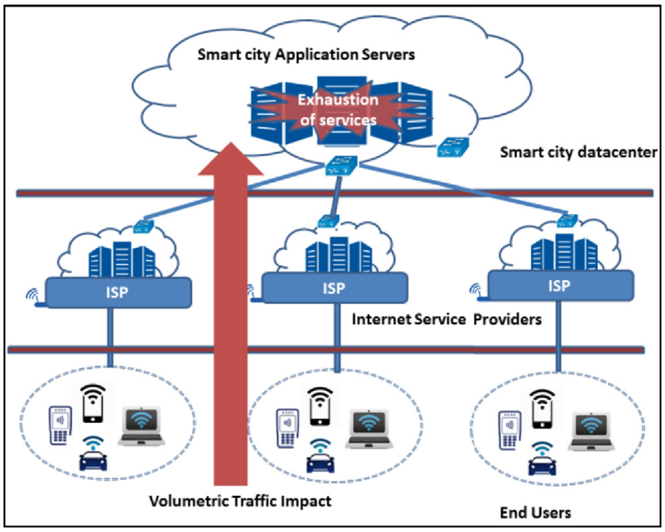
Smart city Application Servers
Exhaustion of services
Smart city datacenter
ISP
ISP
ISP
Internet Service Providers
Volumetric Traffic Impact
End Users

Fig. 2. Attack on smart city application server.

3.1.2. *Attack on smart city application servers*

In an attack on smart city application servers, attackers generate huge amount of malicious requests to a particular application server through compromised computers. Typically, the source address of packets is spoofed in such attacks. The volumetric attack has the potential to overflow the available bandwidth and exhaust the system resources. The aim of the adversary is to saturate the resources of an application server ultimately making services unavailable for legitimate users as shown in Fig. 2.

4. The SEAL framework

Traditional networking approaches have shown there limitation not only in meeting the requirements of modern data centers but also in providing the adequate cybersecurity solutions(Lee et al., 2017)(Kreutz et al., 2015). Typically, the traditional networking involves manual configuration of devices which is both cumbersome and error-prone. Also, it cannot fully utilize the capability of physical network structure as it lacks the global view of network(Wenfeng Xia et al., 2014). Moreover, the increasing number of cyber-attacks, specifically DDoS attacks, substantiate that the existing proprietary and open source solutions for DDoS attack detection and mitigation are not sufficient(Yadav et al., 2016)(Saied et al., 2016). Many such incidents clearly shows that there is a need for new approaches to counter the DDoS attacks(Bawany et al., 2017).

Subsequently, the complexity of smart city datacenter necessitates

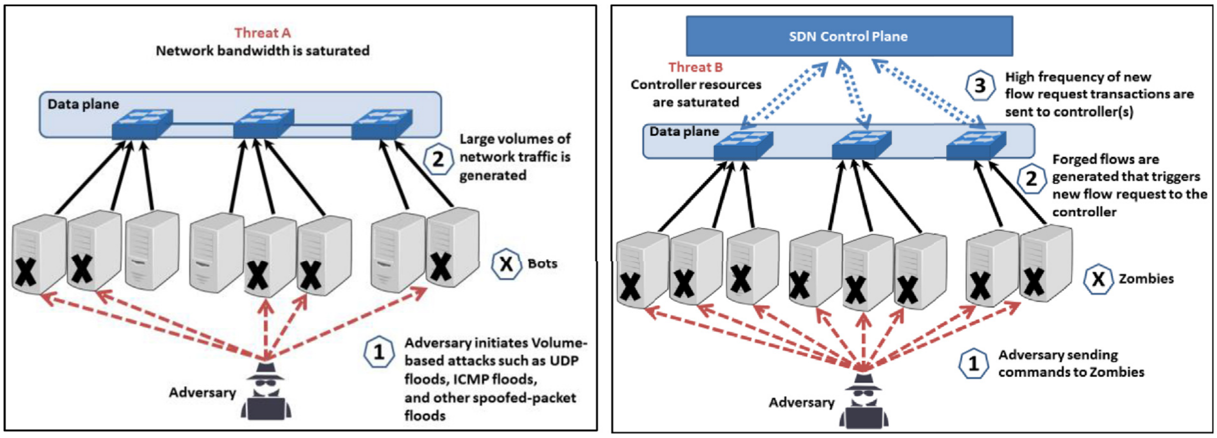
Threat A
Network bandwidth is saturated
Data plane
2 Large volumes of network traffic is generated
X Bots
1 Adversary initiates Volume-based attacks such as UDP floods, ICMP floods, and other spoofed-packet floods
Adversary
SDN Control Plane
Threat B
Controller resources are saturated
3 High frequency of new flow request transactions are sent to controller(s)
Data plane
2 Forged flows are generated that triggers new flow request to the controller
X Zombies
1 Adversary sending commands to Zombies
Adversary

Fig. 1. Threat A and Threat B depicts attack on data plane and control plane respectively.

novel approach in which complexity can be managed seamlessly. The SEAL framework is a customizable DDoS attack detection and mitigation solution that caters the need of application specific security. The SEAL maximizes the bandwidth utilization by implying agile and adaptive solution which reacts to real-time network traffic. Adaptability in SEAL is achieved through implementing a novel adaptive filter based on estimated weighted moving average (EWMA) filter. The use of EWMA filter is motivated by similar work reported in (Kim, 2001)(Lopez and Duarte, 2015)(Shamsi and SyncProbe, 2007)(Shamsi and Brockmeyer, 2012). The Adaptive Filter module of SEAL; monitors the network traffic and subsequently generate attack alerts based on the security requirement of smart city applications and services. The SEAL framework utilizes the classification presented in Table 3 to incorporate adaptive network management scheme, which can estimate the network traffic accordingly.

The SEAL has been designed to meet the demands on the security, flexibility and agility of a smart city applications and infrastructure. The SEAL exploits the key feature of SDN that is, decoupling of control plane and data plane to provide efficient DDoS attack detection and mitigation. Taking advantage of the inherent features of SDN, such as, programmability, global network status collection, centralization, SEAL is capable of providing an effective DDoS attack protection mechanism desired by smart city applications. Moreover, SEAL has a potential to handle scalability and reliability issues with its fault tolerant and resilient design.

The SEAL framework comprises of three modules namely, D-Defense, C-Defense and A-Defense as shown in Fig. 3. These modules collectively ensures the continuity and availability of smart city services by providing the DDoS attack defense mechanism at various levels. SEAL is capable of effectively detecting and mitigating DDoS attacks not only on application servers but also on network resources. The framework is unique in this regard that it provides application specific DDoS attack security solution instead of static threshold mechanism. Moreover, inherently distributed architecture of SEAL ensures fault tolerance, scalability and reliability of the smart city. The SEAL runs on top of the control plane and communicates with the controller via northbound API. Following sections explains the implementation and working of each of the modules in SEAL.

4.1. The D-Defense Module

The D-Defense module serves as a first line of defense against DDoS attacks on smart city applications as it detects and mitigates attack on data plane layer. The D-Defense module has been designed in a way that it can address the application specific security requirements of smart city applications using adaptive solution.

Accordingly, D-Defense module comprises three types of filters to cater the needs of each smart city application category. Adaptability in D-Defense is achieved through tailored version of adaptive EWMA filters which monitor network traffic and subsequently generate attack alerts based on the security requirement of smart city applications and services. This module computes dynamic threshold in real-time for various applications using customized version of EWMA filters.

The D-Defense module has five major components; namely: Traffic Flow Collector(TFC), Policy Engine (PE), Adaptive Filters, Attack Detector and Attack Mitigation Engine as illustrated in Fig. 4. The functions of each of the component- TFC, PE, Attack Detector and Attack Mitigation Engine are decoupled. Therefore, these components can be adapted to future security requirements. The flow entries of OpenFlow switches are collected by Traffic Flow Collector (TFC). Attack detection and mitigation policies can be defined using the PE. Various filters can be selected from Adaptive Filters component to detect an attack. These filters can be configured and activated using PE.

The TFC forwards all the necessary flow related statistics to Adaptive Filter. The Adaptive Filter component estimates the network traffic and forwards the statistics to Threshold detector for further action. Threshold Detector module raises the attack alert and invokes the appropriate mitigation strategy as defined by the Policy Engine.

4.1.1. Adaptive filter

The adaptive filter module is a key module that improvises dynamic threshold mechanism to fulfill the needs of application specific security requirements. The method makes use of the novel adaptive filters based on EWMA filters reported in (Kim, 2001)(Lopez and Duarte, 2015)(Shamsi and Brockmeyer, 2012). The adaptive filter module of D-Defense adjusts the attack detection mechanism to match the applications' security requirements using Equation (1).

Typically, these filters are either able to react swiftly in case of attack or respond sluggishly, depending upon the value of α known as the gain. Lower values for α will generate attack alert instantly, if current traffic rate crosses the threshold. Table 4 shows three different variants of such filters proposed in this research for smart city applications in critical, moderate and low impact categories. These are:

i. Proactive filter
ii. Active filter
iii. Passive filter

4.1.1.1. Estimated traffic calculation for Proactive Filter

$$ET_t = \alpha T_{avg} + (1-\alpha)T_t + \Phi$$
$$\Phi \begin{cases} \Phi, & \Phi > 0 \\ 0, & \Phi \le 0 \end{cases} \tag{1}$$

$$\Phi = \delta\,(T_t - T_{t-1}) \tag{2}$$

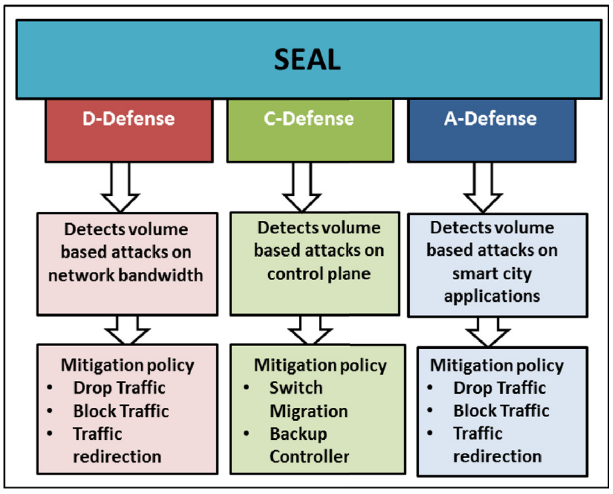

Fig. 3. The SEAL framework.

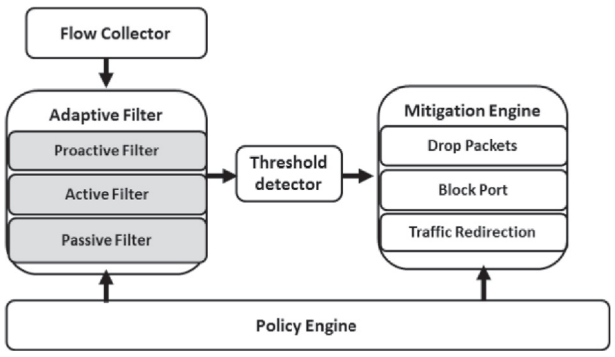

Fig. 4. Design of the D-Defense module.

Table 4
Application security requirements and appropriate filter.

Security Requirements	Appropriate Filter
Extremely Agile: The security solution must take the proactive approach, as the impact of attack is disastrous. Attack detection must have low false negative rate	Proactive Filter
Agile: The security solution is expected to be pragmatic as the impact of attack is critical. An adequate solution that neither reacts before time nor delays the alert is required	Active Filter
Moderately agile: The security solution must take the reactive approach as the continuity of services is more important. Attack detection must have low false positive rate	Passive Filter

$$T_{avg} = \frac{1}{n} \sum_{t=t-(n-1)}^{t} T_t$$

where:
ET_t = Estimated traffic at time t.
T_t = Traffic at time t.
α = gain.
Φ = bias

4.1.1.2. Proactive Filter. In case of the Proactive Filter, $\alpha = 0.1$ in Equation (1) and $n = 2$ in Equation (3) which makes the effect of current traffic rate to be dominant. The Proactive Filter generates security alerts instantly, even before the network traffic reaches the threshold. This filter is suitable for highly critical systems as it provides an early warning mechanism. Consequently, to generate an attack alert before the actual attack goes critical and bypass the threshold, the increasing network traffic variation is required to be aggressively considered. The 'Φ' is defined as a function of network traffic variation that is considered only for Proactive filter.

Proactive Filter satisfies the need of critical applications that require attack alert to be triggered immediately. This filter is designed to identify any escalating trend in network traffic and predict an attack. The filter triggers an attack alert before the network traffic exceeds the predefined threshold. That is, it does not allow any violation of network traffic threshold. This filter may also generate false positive alerts as slight bursts in network traffic triggers attack alert.

4.1.1.3. Active Filter. An Active Filter satisfies the need of applications that expects marginal network traffic bursts. These applications require that attack alert is not triggered instantly on violation of network traffic threshold. The filter suits the application where spikes in network traffic are common.

In case of an Active Filter, $\alpha = 0.5$ in Equation (1) and $n = 4$ in Equation (3) which takes into account both, the current and previous rate of network traffic. The Φ is the difference of network traffic between two consecutive time intervals, current network traffic minus previous network traffic. Value of Φ is calculated using Equation (2).

For most applications this filter will be appropriate as it equally considers both current and previous network traffic rates. Using this filter, little spikes in network traffic can be smoothed out. In Active filter, impact of traffic variations is not needed explicitly; hence the value of 'Φ' is zero.

4.1.1.4. Passive Filter. Subsequently, Passive Filter satisfies the need of application that frequently experiences legitimate network traffic bursts for some period of time, such as in case of flash crowds (Yu et al., 2012). This filter triggers the attack alert only if network traffic remains above the threshold level for extended periods. Therefore, the applications running on network infrastructure that can sustain traffic above the threshold for certain period, and need lowest level of false positives, prefer Passive Filter.

The Passive filter is the most stable filter that reacts gradually to sudden surge in network traffic it takes on the value of $\alpha = 0.9$ in Equation (1) and $n = 8$ in Equation (3). This filter is beneficial where cost of false positive is higher. This filter is reduces the false positive rate and suits the applications that require less agility in security alerts. This filter is suitable for the smoothing bursty traffic with extended burst periods. In the next section, we describe details of experiments conducted to evaluate the D-Defense module.

4.1.2. Experiment design

Mininet (Team, 2012) is used to emulate the network. Mininet emulated network is created on a machine having Intel Core i7, 3.67 GHz 8 GB RAM running Ubuntu. Mininet allows creating realistic virtual networks and experiment with OpenFlow switches, controllers and applications. It is widely used by researchers to create test beds for SDN based networks (Seugwon Shin et al., 2013a,b) (Qazi et al., 2013). A fat tree topology that comprises 64 nodes and 21 OpenFlow switches with each link having a bandwidth of 100 Mb/s is used. The attack is launched on a web server that is hosting a smart city application.

After initializing the network and enabling the D-Defense system 3 min long attack traffic is injected along with legitimate traffic. Legitimate traffic is generated using Distributed Internet Traffic Generator (D-ITG) (Avallone et al., 2003). D-ITG is capable of generating IPv4 and IPv6 traffic replicating the workload of Internet applications. DDoS attack traffic is generated using Iperf (Tirumala et al., 2005).

To make the tests more realistic, real-world experimental scenarios and setups are used. The legitimate traffic and attack traffic is generated simultaneously. The legitimate traffic is generated at the rate of 30 Mb/s and comprises of 85% TCP, 12% UDP and 3% ICMP. The attack traffic ranging from 20 Mb/s to 80 Mb/s is generated. Network traffic threshold is fixed at 60 Mb/s, that is, 60 percent of the network bandwidth is occupied. When the network traffic crosses this predefined threshold it is considered as an attack. This is a realistic threshold obtained by evaluating the application server and network characteristics(Santanna et al., 2015)(Akella and Xiong, 2014). Three types of network traffic models are used that represents each type of smart city application as given in Table 3. The network traffic models used are (a) Poisson Model (p-m), (b) Bursty Traffic model (bt-m) and (c) Bursty Traffic with long burst periods model (btl-m). All three traffic models are depicted in Fig. 5, Fig. 6 and Fig. 7. All three filters are enabled one after the other on each type of network traffic model to evaluate the efficacy of solution. Experiments are conducted using four schemes. In Scheme 0, D-Defense is not activated and fixed threshold is used to detect attacks. In Scheme 1, Scheme 2 and Scheme 3 D-Defense module is activated and Proactive Filter, Active Filter and Passive Filter is used to detect attacks respectively for all traffic models.

4.1.3. Results and discussion

The results of static threshold, Proactive Filter, Active Filter and Passive Filter is depicted in Fig. 9. The experimental results clearly show that Proactive filter is reacting vigorously, especially when there is a significant change in network traffic. Proactive filters react to traffic spikes robustly. Passive Filter acts as the most stable filter.

The static threshold had high false negative rate, ranging from 30% to 40% for all types of traffic. Accuracy is 80% for p-m traffic, however for bt-m and btl-m it reduced to 68% and 63% respectively.

The Proactive Filter is able to bring down the false negative rate to 0% for all types of traffic, thereby fulfilling the need of critical applications that need high degree of protection. The accuracy of Proactive Filter is not at par as it has a high false positive rate ranging from 30% to 53%. Nevertheless, it remains an effective filter due to its ability detect attack before the network traffic crosses the threshold.

The Active Filter works best with the bt-m traffic. The filter is able to detect attacks with 90% accuracy. False positive rate is 7% and false negative rate is 25%. This filter suppresses the transient traffic bursts and does not generate attack alerts on spikes. This filter satisfies the need of

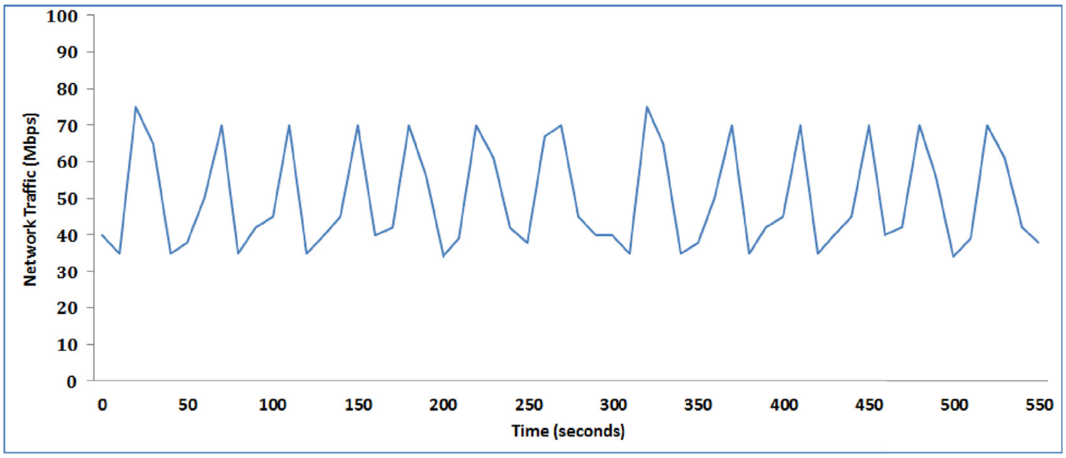

Fig. 5. Poisson traffic model.

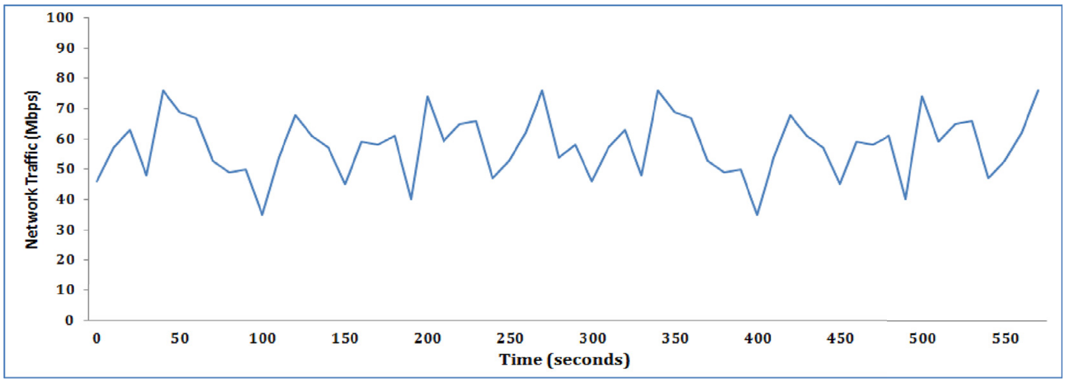

Fig. 6. Bursty Traffic model.

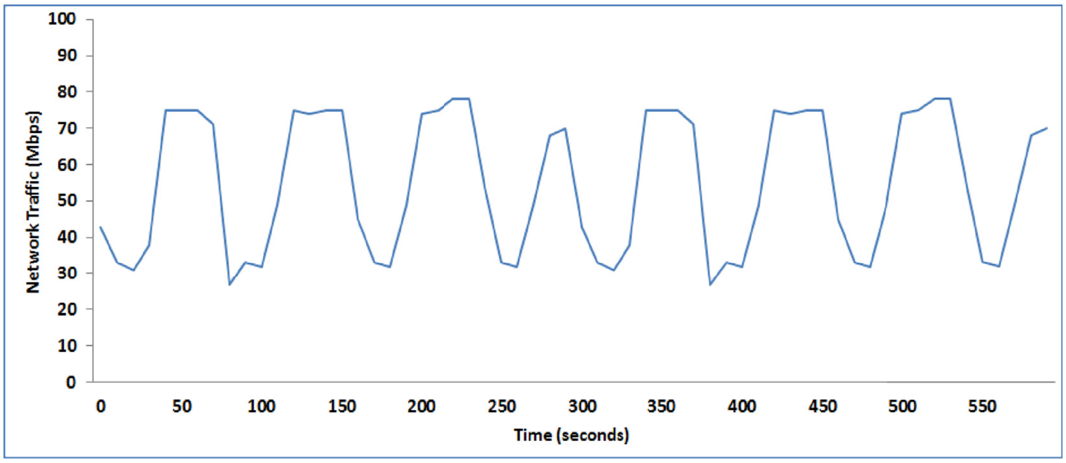

Fig. 7. Bursty Traffic with long burst periods.

applications that expect bursty traffic.

The Passive Filter is the most stable filter. It only generate attack alert when the traffic remains above the threshold for extended time. The extended time can be defined by configuring the value n in equation (1). The filter is suitable for btl-m traffic patterns as it maintains the accuracy of 93%.

4.2. A-Defense Module

The A-Defense module is primarily designed to detect potential DDoS attack on smart city applications with respect to their security requirements as described in Section 3. These requirements are dependent on expected application traffic properties, network and device tolerance. This module is customizable such that it can provide early detection for critical applications that needs extremely agile security solution. Similarly, A-Defense can be configured to suffice the needs of applications that need less agile solutions. Nonetheless, the objective of the A-Defense module is detect and mitigate an attack before the server is completely swamped with malicious packets. To accomplish this goal, A-Defense uses fast, effective and lightweight entropy based mechanism to detect an attack. The A-Defense module is not only capable of detecting a DDoS attack on a particular server but can also implement the predefined mitigation strategy. The implementation of A-Defense module is inspired by the work of (Oshima et al., 2010) (Mousavi and St-Hilaire, 2015).

Entropy based approach has been widely used in traditional networks for detecting network anomalies. This approach has been largely preferred because of its capability of handling large amount of network traffic data in real-time and with low computing overhead. Motivated by the work of Mousavi et al. (Mousavi and St-Hilaire, 2015)(R. Wang et al., 2015), the A-Defense module uses an effective entropy based method to determine the randomness in network traffic flows and detect a potential DDoS attack. Moreover, the flow based nature of SDN complements the entropy based method for detection of DDoS attacks.

DDoS attacks are typically launched by bots having spoofed IP addresses. In case of an attack on a particular server, number of flows having destination IP addresses of the victim server increases. The A-Defense module uses the entropy of destination IP addresses as measure to detect an attack. Typically, in case of DDoS attack request for target server appears more frequently within a short time period, essentially decreasing the entropy of destination IP addresses. The entropy is computed for each window resulting into series of entropies. The window size is defined as the number of packets in each window. When all packets within a window are intended for a same host, minimum entropy occurs. Similarly, when all packets with in a window are intended for unique hosts, maximum entropy occurs. A predefined entropy threshold is set up such that any value less than this predefined threshold is considered an attack.

The A-Defense module comprises five components namely, Metric Collector, Entropy Calculator, Threshold Detector, Policy Engine and Mitigation Engine. Fig. 8 depicts the logical structure of A-Defense module. The Metric Collector component collect the statistics periodically, and keeps the count of number of flows with same destination address within a window size. When there is an attack, number of flows having the same destination IP addresses increases. The Entropy Calculator component use the frequency of each destination IP address within a window size, to estimate its probability.

Next section describes the experimental setups that are used to investigate the feasibility of the SEAL's A-Defense module.

4.2.1. *Experiments and results*

Several experiments are designed and conducted to evaluate the performance and efficiency of A-Defense in protecting smart city applications from DDoS attacks. The testbed is setup on a machine having Intel Core i7, 3.67 GHz 8 GB RAM running Ubuntu.

For evaluation, Mininet is used, as the network emulator, with a Java-based open source distributed controller, ONOS. The experimental testbed has been setup after carefully reviewing the work of many researchers (David and Thomas, 2015)(R. Wang et al., 2015)(Mousavi and St-Hilaire, 2015). The dataset used for initial experiments is a subset of ISCX dataset. The dataset is split into small windows consisting of defined number of packets. Experiments are conducted to study the impact of window size on entropy. The size of windows used for experiments is 100

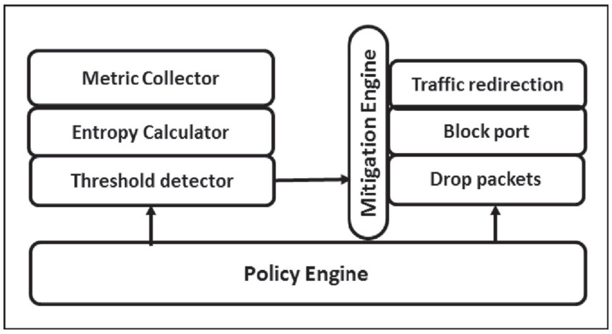

Fig. 8. Design of the A-Defense module.

packets, 200 packets, 300 packets, 500 packets and 1000 packets. The entropies are calculated for both, attack dataset and no attack dataset, as shown in Table 5. It is observed that mean entropy is higher in case of no attack for all window sizes. The calculation method used to find the suitable threshold is similar to one suggested in (Mousavi and St-Hilaire, 2015). After running simulations number of times, it is observed that number of false positive increases as window size is decreased. It is also observed that number of false positive decreases as the window size is increased. The comprehensive results of the experiments are summarized in Tables 5 and 6. Hence, relationship between window size and number of false positives can be represented as given below:

$$\textit{Window size}\ \alpha \frac{1}{\textit{Number of False Positives}}$$

The observation shows high sensitivity and specificity for all window sizes. Small window size tend to detect an attack earlier, however it may result in high rate of false positive cases comparatively. Conversely, large window size results in less number of false positive cases though it may take longer to detect an attack.

Nevertheless, small window has a greater computational overhead.

To reduce the FPR in small window size, the experiments are repeated and attack alert is fired only when the entropy value remained below threshold for ‘x’ consecutive windows. The value of ‘x’ is varied from 2 to 4. The best results are obtained when the value of ‘x’ is 3. The results are presented in Table 7.

Taking into account the above mentioned factors, application specific requirements can be met varying the window size, and number of consecutive windows as given in Table 8. As discussed in Section 3, critical applications needs low false negative rate, hence, the small window size suits critical application requirements. However, less critical applications can afford to delay the attack detection therefore large window size is appropriate for such applications. These window sizes has been termed as Proactive, Active and Passive in accordance with the type of application.

To examine computing overhead of A-Defense module, the CPU usage graph is monitored. Initially, the A-Defense application services are stopped and the attack traffic is sent to application server. The scenario is repeated after turning on the A-Defense services. The difference of CPU and memory usage is found to be insignificant. The solution is not affected by the type of attack traffic as the calculation is based on destination IP address.

4.3. *C-defense module*

In the SEAL framework, C-Defense module is dedicated for monitoring control plane. This module actively checks the load on each controller and immediately detects any activity leading to possible DDoS attack on SDN controller.

SDN brings about a major revolution in the networking by splitting the control and data plane. This core feature of SDN provides several benefits. The control plane has the capability to control all physical devices of the network. The control plane is logically centralized and is incharge of all forwarding decisions. The control plane becomes the most critical part of the whole network making its reliability, scalability and availability a major concern (Levin et al., 2012)(Yeganeh et al., 2013)(Karakus and Durresi, 2017)(Hu et al., 2018). The control plane must not be a single point of failure, therefore the security and reliability of control plane is of immense importance.

SDN supports dynamic network policy implementations instantaneously across the network. Real-time decisions can be taken and new flow rules can be installed on the OpenFlow switches by the controller (Astuto et al., 2014). Typically, whenever a packet arrives at an OpenFlow switch it is forwarded as per rules defined in the forwarding table. However, if the packet does not match any entry in the flow table it is forwarded to controller. The controller may install a new rule in a flow

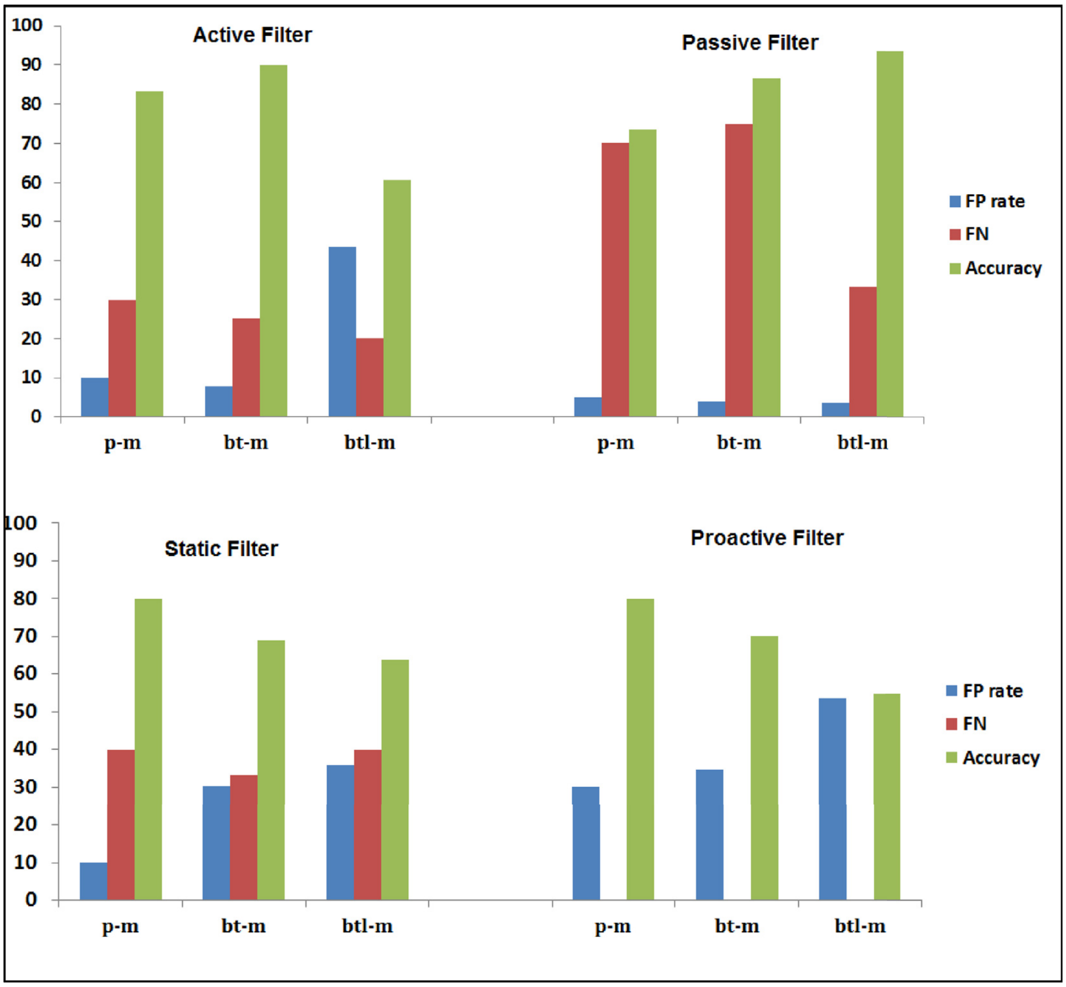

Fig. 9. Experimental results of all filters.

Table 5
Mean Entropy (ME) and Standard deviation(SD) calculation for different window sizes.

Window size	ME No Attack dataset	SD No Attack dataset	ME Attack dataset	SD Attack dataset	Threshold
100	3.38	0.98	1.87	0.90	3.25
200	3.71	1.06	2.01	0.94	3.55
300	3.90	1.10	2.10	0.97	3.73
500	4.14	1.15	2.20	1.00	3.93
1000	4.44	1.26	2.32	1.05	4.14

Table 6
False Positive Rate (FPR), False Negative Rate (FNR), Accuracy, Sensitivity and Specificity for different window sizes.

Window size	FPR	FNR	Accuracy	Sensitivity	Specificity
1000	9.43	4.61	94.44	97.64	82.76
500	16.04	3.92	93.7	96.08	83.96
300	17.42	2.76	94.35	95.79	88.02
200	18.52	1.38	95.21	95.54	93.62
100	22.99	0.46	95.17	94.74	97.57

table of OpenFlow switch and following packets are forwarded through the rule installed. Although, this capability of SDN leads to many advantages such as programmability, flexibility and runtime updates of network polices, the attackers can easily target SDN controllers by sending a lot of forged packets that cannot be matched by the flow entries, in OpenFlow switches. Moreover, in a large scale network number of new flow requests may increase drastically that can degrade the performance of a controller or lead to control plane failure.

Table 7
FPR, FNR, Accuracy, Sensitivity and Specificity for different window sizes when ‘x’ is 3.

Window Size	FPR	FNR	Accuracy	Sensitivity	Specificity
1000	1.89	6.91	94.07	99.51	77.61
500	9.43	5.06	94.09	97.64	81.36
300	14.04	3.04	94.79	96.56	87.43
200	12.96	1.38	96.31	96.84	94
100	13.41	0.55	96.96	96.86	97.41

Table 8
Window size for application specific security.

Category	Security Solution Requirement	Window Size
Critical	Extremely Agile	Proactive (100–200 packets)
Moderate	Agile	Active (300–500 packets)
Low	Moderately agile	Passive (1000 packets)

Hence many distributed control plane architectures have been proposed (Bannour et al., 2018). However, only few controller platforms, such as, DISCO (Phemius et al., 2014), Kandoo (Hassas Yeganeh et al., 2012), HyperFlow (Tootoonchian and Ganjali, 2010) and ONIX(Koponen et al., 2010) are focused on resiliency aspect. Among these controller platforms Open Networking Operating System (ONOS) (Berde et al., 2014), offers an open-source and stable implementation(Gerola et al., 2015).

ONOS is a distributed Network Operating System which provides a logically centralized view of the network. Though, ONOS improves the availability and scalability of the controller by using multiple controllers, it lacks a flexible mechanism to balance load among controllers (Li et al., 2016) (J. J Yu et al., 2016). Due to this, one controller can be overloaded causing performance degradation, in case of legitimate excessive load or DDoS attack.

To overcome this limitation, the C-Defense module has been proposed and implemented. This module is designed to meet the resiliency, scalability and high availability requirements, for SDN based smart city network. This modules collects the load of all controllers periodically and use filters for estimating the load. In the event of controllers' estimated load crossing the threshold, mitigation strategy is activated.

The C-Defense module incorporates load balancing as an initial DDoS attack mitigation strategy. However, it is also capable of identifying the OpenFlow switch responsible for overloading the controller and blocking the excessive traffic.

Load balancing ensures consistent performance of the controller by distributing its load to other controllers, in a cluster, before the controller reaches its capacity. It provides a fault tolerance mechanism as each device registers to multiple controller instances. The control is immediately transferred to back up controller in case of a fault. The change of control is notified throughout the network using distributed registry. Similarly, data plane is also resilient, whenever a link or a device fails traffic is rerouted automatically. With C-Defense, ONOS cluster is able to operate with high bandwidth and low latency.

The design of C-Defense module is depicted in Fig. 10. It consists of many small components that assists and coordinates in monitoring and managing the load of controller. The Policy Engine keeps the record of controllers' predefined threshold. The predefined threshold is defined as a limit after which the performance of controller may deteriorate. The Load Information Collector component periodically collects the load metrics and load calculation component checks whether the controller's load had reached the threshold or not.

The Balance Decision component takes the load balancing decision, such as, choosing lightly-loaded controllers on which switches can be migrated. The balance decision component implements the algorithm given in Fig. 11.

The adaptive filter implements two filters- Proactive Filter and Active Filter. Typically, these filters are either able to react swiftly, as the load on the controller reaches the threshold, or respond slowly, masking the transients depending upon the value of α as discussed in Section 4. Lower values for α will lead to immediate load balancing request and higher values of α will differ the load balancing request.

4.3.1. *Attack Detection and load balancing algorithm*

Algorithm given in Fig. 11, detects DDoS attack on controller by

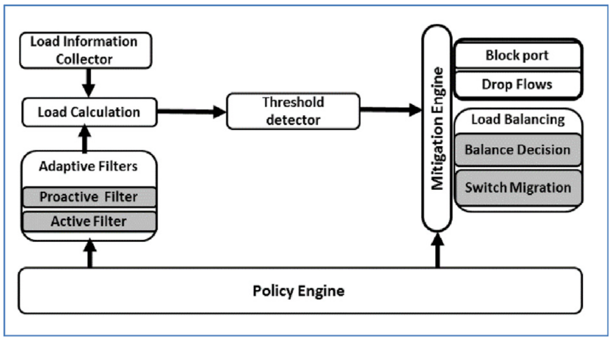

Fig. 10. Design of the C-Defense module.

identifying the switch that is sending highest amount of control traffic. The algorithm is designed to mitigate an attack by load balancing and/or blocking the excessive control traffic from switch. In case of load balancing, key objective is to balance the control traffic load among multiple controllers, such that number of switch migrations remain minimum. To achieve this objective, switches are migrated from most over loaded controller to least loaded controller. Also, those switches are selected for migration that are sending highest amount of control traffic. Number of migrations allowed by the switch is defined by the constant MAXMIG. This constant can be reset periodically. When the same switch is migrated, maximum allowed number of times, within a certain time period, the mitigation mechanism is invoked that blocks the excessive traffic from switch.

Although, it is assumed that there will be many controllers in the cluster to ensure that the consistent performance of control plane. However, in case, no controller is available for load balancing the excessive traffic will be blocked to mitigate an attack.

In algorithm given in Fig. 11, *G* represents the topology of the network. It represents the controller to switch connectivity. *Tx* is an aggregated control traffic received by the controller *Cx* from all switches associated with it. *M* denotes the mastership of controller to switches, that is, each element *M1,M2, ….Mx* consists of a controller and a list of switches associated with it. The subscript denotes the controller.

4.3.2. *Experiments and results*

DDoS attack is launched on controller by sending a large number of new flow requests. As OpenFlow switches are unable to recognize these new flows, these requests are sent to SDN controller for computing a path for new flows.

Consequently, these requests increase control traffic at the controller leading to performance degradation or complete denial of service.

The C-Defense module, is designed to ensure that one controller within a cluster, is not overloaded and the load is distributed among other controllers. This module monitors the state of all OpenFlow switches and controllers in a cluster. As, the threshold of any controller is reached, the OpenFlow switch that is generating maximum control traffic to the controller, is migrated to other controller with least load. The C-Defense module is capable of encountering DDoS attack either by load balancing or blocking the excessive control traffic.

To determine the single controller's capacity of processing maximum control traffic the stress test is performed. It is observed that controller halted at the control traffic reached around 9500 kbps. The capacity and performance of controller is dependent upon the configuration of physical machine on which the controller is running. The experiment is repeated several times after which 7000 kbps is chosen as threshold value for further experiments. Nonetheless, the threshold value can be configured using the Policy Engine of C-Defense whenever required. After the threshold has been determined, control traffic on controller is increased to emulate a DDoS attack. In these experiments, control traffic is considered as a load of controller. However, physical characteristics of machine, such as memory and processor usage can also be included in the calculation of load.

To evaluate the C-Defense module, experiments are performed with and without C-Defense activation. In Scheme 0 experiment, OpenFlow switches are equally distributed to controllers and C-Defense is not installed. In Scheme 1 experiment, C-Defense is enabled on ONOS controllers.

To conduct the experiments a test bed is created using Mininet emulated network connected to ONOS cluster having three controllers. Each controller is running on a machine having Intel Core i7, 3.67 GHz processor and 4 GB RAM running Ubuntu. Mininet emulated network is created on a similar machine having Intel Core i7, 3.67 GHz processor and 4 GB RAM running Ubuntu. The C-Defense module is deployed on ONOS cluster and D-ITG tool is used to generate traffic. The fat tree topology comprising of 21 OpenFlow switches, and 64 hosts is created to carry out the experiments.

Algorithm: Attack detection and load balancing algorithm

```
input   $\mathcal{T}$: Aggregated control traffic received at controller
        $\mathcal{E}$: Estimated aggregated control traffic received at controller
        $\beta$ : control traffic from switch to controller
        $\mathcal{M}$: Controller to Switch mapping
        $\mathcal{G}$: Topology
        $\mathcal{C}_x$: Controllers
        $S_y$=Switches
        $x$: Number of controllers
        $y$: Number of switches
        $f$: Type of filter activated
        $\tau$: Threshold
        $\mathcal{P}$: Policy
        $\omega$: Over Placement avoidance constraint

while TRUE do {

    [$\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3 \ldots . \mathcal{T}_x$]= controllersLoadMeasurement( )
    [ $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \ldots . \mathcal{E}x$] = estimatedControllerLoad(Filter $f$, [$\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3 \ldots . \mathcal{T}_x$])
    $\mathcal{E}_{max}$ = max ([ $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \ldots . \mathcal{E}x$])
    $\mathcal{E}_{min}$ = min ([ $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \ldots . \mathcal{E}x$])
    balanceFlag=FALSE;

    if ($\mathcal{E}_{max}$ > $\tau_x$) {
        balanceFlag=TRUE;
        [$\beta_1, \beta_2, \beta_3 \ldots . \beta_n$]= SwitchToControllerTraffic($\mathcal{C}_{max}$,$\mathcal{M}_{max}$)
        sortDescending($\beta_1, \beta_2, \beta_3 \ldots . \beta_n$)
        //n is the number of switches connected to $\mathcal{C}_{max}$
        for(count=1;count<n+1;count++)  {
            if (( $\beta_{count}$ + $\mathcal{E}_{min}$ + $\omega$) < $\tau_y$) AND isConnected($S_{count}$,$\mathcal{C}_{min}$,$\mathcal{G}$) {
                if ($S_{count}$ .getMigrated( )< MAXMIG) {
                    migrate($S_{count}$ ,$\mathcal{C}_{min}$)
                    $S_{count}$ .setMigrated(++getMigrated( ))
                    }
                else
                    mitigate($S_{count}$, $\mathcal{P}$)

            balanceFlag=FALSE
            break
            }
        }
    }
    delay(1000) // in milli-seconds
}
```

Fig. 11. Attack detection and load balancing algorithm.

Scheme 0: C-Defense module not activated

In the first experiment, OpenFlow switches are equally distributed among the controllers and each controller had 7 switches. Fig. 12 demonstrates the control traffic load on each controller. Control traffic on one controller is increased gradually, but the load is not distributed to other controllers. Therefore, the performance of controller C3 deteriorates, while other two controllers in a cluster remains lightly loaded. In this case, control traffic load is not considered naïve switch to controller placement is implemented that only distributes the number switches. Therefore, this module fails in balancing the load among controllers. This strategy does not provide any protection against DDoS attacks.

Scheme 1: C-Defense module activated.

In this experiment, the C-Defense module is activated on the controller. Initially each controller is assigned same number of switches. However, load on each controller is continuously monitored by C-Defense module. As the load on controller C3 increases and crosses the threshold, the switch migration takes place. The switch s1, that is generating the highest amount of control traffic, is selected for migration. Controller C2 is the least loaded controller in the cluster, hence s1 is migrated to controller C2 as shown in Fig. 13.

To further evaluate the mitigation capabilities of C-Defense module, the control traffic from s1 is continuously increased. This control traffic from s1 overloaded the controller C2 also. As maximum allowed migrations for switch is set to 1, hence further migrations are not allowed and the control traffic from s1 is blocked as shown in Fig. 14.

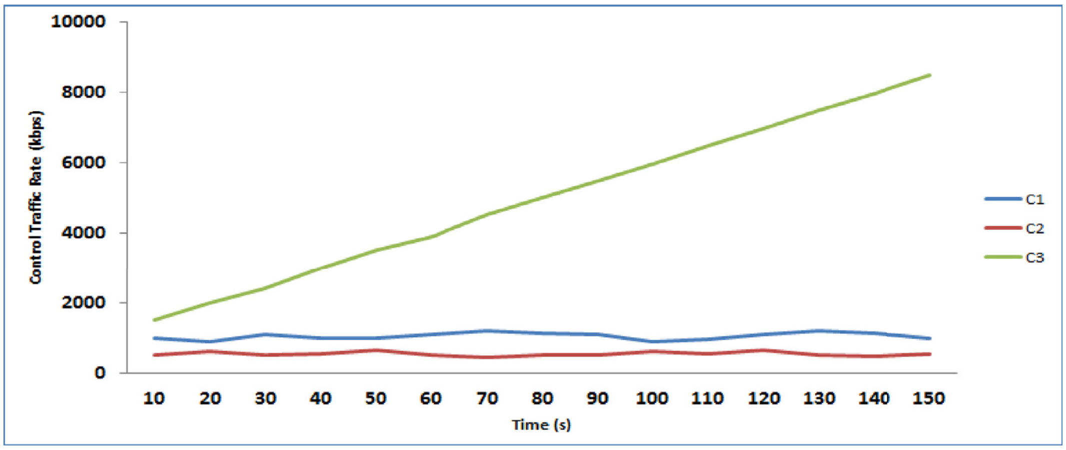

Fig. 12. Control traffic on controllers with base case.

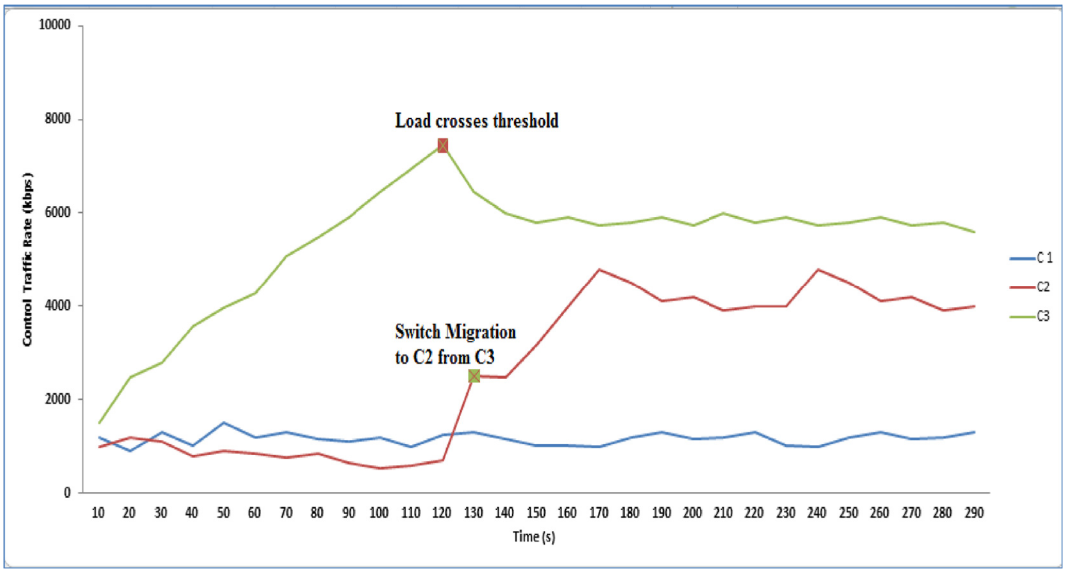

Fig. 13. Load balancing with switch migration.

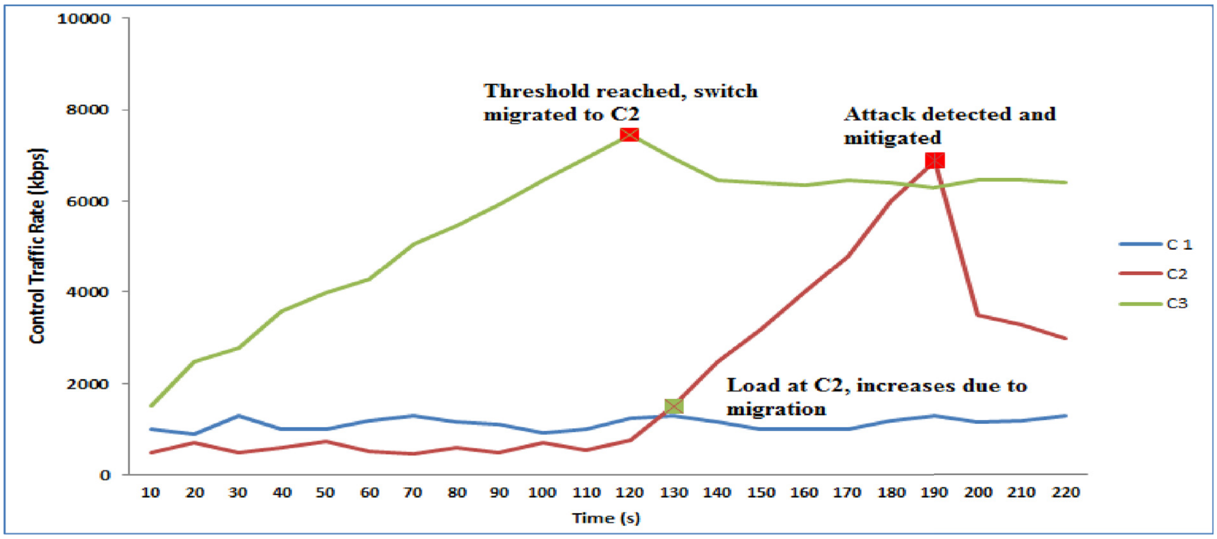

Fig. 14. Load balancing, attack detection and mitigation.

5. Conclusion

This research is motivated by the key fact that DDoS attacks can be a major barrier to sustainable services of smart cities. Precisely, the primary goal of this research is to provide an adaptive and effective DDoS attack prevention mechanism for smart city services and applications. The key contribution of the SEAL framework is to incorporate an application specific security criteria to detect DDoS attacks by using dynamic threshold levels. This feature is implemented using customized version of EWMA filters that predicts the occurrence of an attack on the basis of real-time network traffic.

The SEAL framework has been specifically designed to meet the adaptability requirements of smart city applications, hence providing an effective protection for smart city applications from DDoS attacks. The SEAL framework provides various type of filters (Proactive Filter, Active Filter, and Passive Filter) to meet the application-specific security requirements. These filters can be customized to achieve essential accuracy of attack detection.

The SEAL framework is an SDN based solution hence it has several advantages over traditional solutions. First, the network policies are defined through software and no low-level device configurations are required. Second, additional hardware is not required as this software solution can be deployed on existing OpenFlow hardware, therefore it is cost effective and has no interoperability issues. Third, network security policies can be updated and enforced at runtime. The detection and mitigation modules can automatically react to changes in the network state and thus implement strong polices, promptly. Fourth, the global knowledge of network topology and devices statistics ensures the effective network-wide protection from DDoS attacks.

The SEAL framework consists of three modules A-Defense, D-Defense and C-Defense that together form a comprehensive protective shield that guards smart city applications along with network infrastructure from DDoS attacks ensuring high availability.

The A-Defense module detects and mitigates DDoS attacks on applications, whereas D-Defense and C-Defense modules are responsible for detecting attacks on data plane and control plane, respectively. The C-Defense module is also capable of providing load balancing among the controllers within a cluster. This feature ensures that controller does not become performance bottleneck and the control plane remains fault tolerant and resilient.

Experimental analysis verifies the effectiveness and efficiency of the SEAL framework and its modules. It has been demonstrated, by conducting various set of experiments, that the framework is capable of detecting and mitigating DDoS attack in real-time fulfilling the security requirements of smart city applications. Extensive experiments confirm that the SEAL framework is capable of adapting to application-specific criteria in generating attack alerts. Specifically, it can protect critical applications by incorporating the Proactive filter. During the course of experiments, network traffic remained below the threshold level when Proactive Filter is activated and attacks are mitigated effectively. Similarly, for moderate-impact and low-impact applications, the Active Filter and Passive Filter serve the purpose of delaying the attack alerts to bring down the false positive rate.

The SEAL Framework is highly effective for smart city applications where the traffic patterns and security requirements are known. However, optimization of filters may be necessary to ensure the effectiveness in other application domains where traffic patterns differ or security requirements are ambiguous. Similarly, the SEAL framework may bot support application areas where source based attack detection is required.

The SEAL framework is not only capable of detecting network-wide attacks but can also respond immediately to attacks by leveraging the centralized control and dynamic network policy enforcement features of SDN. Moreover, this framework is resilient and fault tolerant solution against DDoS attacks.

To examine the computing overhead of the SEAL framework, the CPU and memory usage graph is monitored for all three modules. It is observed that the CPU and memory usage for all modules is insignificant. However, as the number of switches connected to a controller increases, performance may be affected. The distributed computing model can be utilized for large networks to ensure the consistent performance of the SEAL framework.

The SEAL framework is not only limited to smart city application. It can be utilized to protect all type of applications and services from DDoS attack. The framework can be deployed on any SDN based network to provide DDoS attack detection and mitigation services across the network. However, smart city applications as been presented as a use-case to prove the effectiveness of the SEAL framework. The framework is highly customizable such that it can meet the security requirement of various applications. Further, this framework has been designed in modular form, therefore the security requirements for each layer can be defined independently.

6. Future work

Another direction in this research is to discriminate legitimate traffic from malicious traffic. A deep packet inspection (DPI) mechanism can be included in the SEAL framework that uses the contents of header fields to search for predefined patterns in the packet (Bouet et al., 2013)(Najam et al., 2015).

The SEAL framework can be extended by including a source trace-back (S. S Yu et al., 2016) mechanism for identifying attackers and locating their distributed sources in real-time. If true origin of the attacker can be identified, it can be used to block further occurrences of attacks from the potential sources (Ahmed et al., 2016).

Moreover, increasing number of cyber-attacks give rise to another key requirement in protecting cyber-physical systems. The cyber security investigators need to analyze network events to in order to determine how an attack was carried out (Li et al., 2017). Various network forensics techniques are used to facilitate tracking internal and external network attacks. An intelligent network forensics tool (Khan et al., 2016) can be incorporated as an additional module in the SEAL framework enabling complete exploration of digital evidence after the occurrence of DDoS attack. This will assist in the identification of the attackers and will reduce the potential risks of further attacks.

Acknowledgements

The work has been supported through Higher Education Commission (HEC), Pakistan, NRPU grant 5946. We are also grateful to the NVIDIA corporation for supporting our research.

References

Akhunzada, A., Gani, A., Anuar, N.B., Abdelaziz, A., Khan, M.K., Hayat, A., Khan, S.U., 2016 Feb 1. Secure and dependable software defined networks. Journal of Network and Computer Applications 16, 199–221.

Ahmed, A.A., Sadiq, A.S., Zolkipli, M.F., 2016. Traceback model for identifying sources of distributed attacks in real time. Secur. Commun. Netw. 9, 2173–2185. https://doi.org/10.1002/sec.1476.

Akella, A.V., Xiong, K., 2014. Quality of service (QoS)-Guaranteed network resource allocation via software defined networking (SDN). In: IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, pp. 7–13. https://doi.org/10.1109/DASC.2014.11, 2014.

Alibasic, Armin, Al, Junaibi, Reem, Aung, Zeyar, Woon, Lee, Wei, Omar, M.A., 2016. Cybersecurity for smart cities: a brief review. In: International Workshop on Data Analytics for Renewable Energy Integration. Springer, pp. 22–30.

Alsmadi, I., Xu, D., 2015. Security of software defined networks: a survey. Comput. Secur. 53, 79–108. https://doi.org/10.1016/j.cose.2015.05.006.

Astuto, B.N., Mendonca, M., Nguyen, X.N., Obraczka, K., Turletti, T., 2014. A survey of software-defined Networking : past , present , and future of programmable. IEEE Communications Surveys & Tutorials 16, 1617–1634. https://doi.org/10.1109/SURV.2014.012214.00180>.

Avallone, S., Pescapè, A., Ventre, G., 2003. Distributed Internet Traffic Generator (D-ITG): analysis and experimentation over heterogeneous networks. In: ICNP 2003 Poster Proceedings, International Conference on Network Protocols. Atlanta, Georgia.

Bannour, F., Souihi, S., Mellouk, A., 2018. Distributed SDN control: survey, taxonomy, and challenges. IEEE Communications Surveys and Tutorials 20, 333–354. https://doi.org/10.1109/COMST.2017.2782482.

Bawany, N.Z., Shamsi, J.A., 2016. Application Layer DDoS Attack Defense Framework for Smart City Using SDN. Computer Science, Computer Engineering, and Social Media (CSCESM), pp. 1–9, 2016.

Bawany, N.Z., Shamsi, J.A., Salah, K., 2017. DDoS attack detection and mitigation using SDN: methods, practices, and solutions. Arabian J. Sci. Eng. 42, 425–441. https://doi.org/10.1007/s13369-017-2414-5.

Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., et al., 2014. ONOS: towards an open, distributed SDN OS. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking - HotSDN '14, pp. 1–6. https://doi.org/10.1145/2620728.2620744.

Bouet, M., Leguay, J., Conan, V., 2013. Cost-based placement of virtualized deep packet inspection functions in SDN. In: Proceedings - IEEE Military Communications Conference MILCOM, pp. 992–997. https://doi.org/10.1109/MILCOM.2013.172.

Braga, R., Mota, E., Passito, A., 2010. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: Proceedings - Conference on Local Computer Networks, LCN. https://doi.org/10.1109/LCN.2010.5735752, 408–15.

Buragohain, C., Medhi, N., 2016. FlowTrApp: an SDN based architecture for DDoS attack detection and mitigation in data centers. In: 3rd International Conference on Signal Processing and Integrated Networks (SPIN), pp. 519–524. https://doi.org/10.1109/SPIN.2016.7566750, 2016.

Cerrudo, C., April 2015. Hacking smart cities. In RSA Conference, San Francisco, pp. 2–18.

Cerrudo, C., IOActive, L., 2015. An Emerging US (and World) Threat : Cities Wide Open to Cyber Attacks 1–20.

Chen, K.Y., Junuthula, A.R., Siddhrau, I.K., Xu, Y., Chao, H.J., 2016. SDNShield: towards more comprehensive defense against DDoS attacks on SDN control plane. In: IEEE Conference on Communications and Network Security, CNS 2016 2017, pp. 28–36. https://doi.org/10.1109/CNS.2016.7860467.

Das, A., Sharma, S.C.M., Ratha, B.K., 2019. The new Era of smart cities, from the perspective of the Internet of things. In: Smart Cities Cybersecurity and Privacy. Elsevier, pp. 1–9.

David, J., Thomas, C., 2015. DDoS attack detection using fast entropy approach on flow-based network traffic. Procedia Computer Science 50, 30–36. https://doi.org/10.1016/j.procs.2015.04.007.

Dharma, N.I.G., Muthohar, M.F., Prayuda, J.D.A., Priagung, K., Choi, D., 2015. Time-based DDoS detection and mitigation for SDN controller. APNOMS. In: 17th Asia-Pacific Network Operations and Management Symposium: Managing a Very Connected World. https://doi.org/10.1109/APNOMS.2015.7275389, 2015:550–3.

Dotcenko, S., Vladyko, A., Letenko, I., 2014. A fuzzy logic-based information security management for software-de fi ned networks. Advanced communication Technology (ICACT). In: 2014 16th International Conference on, pp. 167–171. https://doi.org/10.1109/ICACT.2014.6778942.

Gerola, M., Santuari, M., Salvadori, E., Salsano, S., Ventre, P.L., Campanella, M., et al., 2015. ICONA: inter cluster onos network application. In: 1st IEEE Conference on Network Softwarization: Software-Defined Infrastructures for Networks, Clouds, IoT and Services, NETSOFT 2015. https://doi.org/10.1109/NETSOFT.2015.7116173.

Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., Maglaris, V., 2014. Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. Comput. Network. 62, 122–136. https://doi.org/10.1016/j.bjp.2013.10.014.

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., et al., 2008. NOX: towards an operating system for networks. SIGCOMM Computer Communication Review 38, 105–110. https://doi.org/10.1145/1384609.1384625.

Han, B., Yang, X., Sun, Z., Huang, J., Su, J., 2017. OverWatch : a cross-plane DDoS attack defense framework with collaborative intelligence in SDN. In: Security and Communication Networks,Hindawi.

Hassas Yeganeh, S., Ganjali, Y., Yeganeh, S.H., Ganjali, Y., 2012. Kandoo: a framework for efficient and scalable offloading of control applications. In: Proceeding HotSDN '12 Proceedings of the First Workshop on Hot Topics in Software Defined Networks, pp. 19–24. https://doi.org/10.1145/2342441.2342446.

Hiller, J.S., Russell, R.S., 2013. The challenge and imperative of private sector cybersecurity: an international comparison. Comput. Law Secur. Rep. 29, 236–245. https://doi.org/10.1016/j.clsr.2013.03.003.

Hu, T.A.O., Guo, Z., Yi, P., Baker, T., Lan, J., 2018. Multi-controller based software-defined Networking . Surveyor 6.

James, Clive, 2016. Cybersecurity opportunities, threats challenges. Australian Cyber Security Journal 72.

Karakus, M., Durresi, A., 2017. A survey: control plane scalability issues and approaches in Software-Defined Networking (SDN). Comput. Network. 112, 279–293. https://doi.org/10.1016/j.comnet.2016.11.017.

Kaufman, C., Perlman, R., Sommerfeld, B., 2003. DoS protection for UDP-based protocols. In: Proceedings of the 10th ACM Conference on Computer and Communication Security - CCS '03, p. 2. https://doi.org/10.1145/948109.948113.

Khan, S., Gani, A., Wahab, A.W.A., Shiraz, M., Ahmad, I., 2016. Network forensics: review, taxonomy, and open challenges. J. Netw. Comput. Appl. 66, 214–235. https://doi.org/10.1016/j.jnca.2016.03.005.

Khondoker, R., Zaalouk, A., Marx, R., Bayarou, K., 2015. Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers Feature-Based Comparison and Selection of Software Defined Networking. SDN) Controllers. https://doi.org/10.13140/2.1.2307.7125.

Kianpisheh, A., Mustaffa, N., Limtrairut, P., Keikhosrokiani, P., 2012. Smart Parking System (SPS) architecture using ultrasonic detector. International Journal of Software Engineering and Its Applications 6, 51–58.

Kim, M.N.B., 2001. Mobile network estimation. In: Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM, pp. 298–309. https://doi.org/10.1145/381677.381705.

Kimani, K., Oduol, V., Langat, K., 2019. Cyber security challenges for IoT-based smart grid networks. International Journal of Critical Infrastructure Protection 25, 36–49.

Kitchin, R., 2016. Getting Smarter about Smart Cities: Improving Data Privacy and Data Security, vols. 1–83.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., et al., 2010. Onix: a distributed control platform for large-scale production networks. In: Proceedings of USENIX Operating Systems Design and Implementation (OSDI).

Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E.C.E., Azodolmolky, S., Uhlig, S., et al., 2015. Software-defined networking: a comprehensive survey. Proc. IEEE 103, 14–76. https://doi.org/10.1109/JPROC.2014.2371999.

Lee, S., Yoon, C., Lee, C., Shin, S., Yegneswaran, V., Porras, P., 2017. DELTA: a security assessment framework for software-defined networks. In: Proceedings 2017 Network and Distributed System Security Symposium. https://doi.org/10.14722/ndss.2017.23457.

Levin, D., Wundsam, A., Heller, B., Handigol, N., Feldmann, A., 2012. Logically centralized? State distribution trade-offs in software defined networks. HotSDN 1–6. https://doi.org/10.1145/2342441.2342443.

Li, J., Yoo, J.-H., Hong, J.W.-K., 2016. Dynamic control plane management for software-defined networks. Int. J. Netw. Manag. 26, 111–130. https://doi.org/10.1002/nem.1924.

Li, Z., Pan, H., Liu, W., Xu, F., Cao, Z., Xiong, G., 2017. A network attack forensic platform against HTTP evasive behavior. J. Supercomput. 73, 3053–3064. https://doi.org/10.1007/s11227-016-1924-3.

Lim, S., Ha, J., Kim, H., Kim, Y., Yang, S., 2014. A SDN-oriented DDoS blocking scheme for botnet-based attacks. In: International Conference on Ubiquitous and Future Networks, ICUFN. https://doi.org/10.1109/ICUFN.2014.6876752, 63–8.

Lopez, M.A., Duarte, O.C.M.B., 2015. Providing elasticity to intrusion detection systems in virtualized Software Defined Networks. IEEE Int. Conf. Commun. 7120–7125. https://doi.org/10.1109/ICC.2015.7249462, 2015–Septe.

Maziku, H., Shetty, S., Nicol, D.M., 2019. Security risk assessment for SDN-enabled smart grids. Comput. Commun. 133, 1–11.

Medved, J., Varga, R., Tkacik, A., Gray, K., 2014. OpenDaylight: towards a model-driven SDN controller architecture. In: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks. https://doi.org/10.1109/WoWMoM.2014.6918985. WoWMoM 2014 2014.

Mehdi, S.A., Khalid, J., Khayam, S.A., 2011. Revisiting traffic anomaly detection using software defined networking. In: International Workshop on Recent Advances in Intrusion Detection. Springer, Berlin, Heidelberg, pp. 161–180. https://doi.org/10.1007/978-3-642-23644-0_9.

Mousavi, S.M.S., St-Hilaire, M., 2015. Early detection of DDoS attacks against SDN controllers. In: International Conference on Computing, Networking and Communications, ICNC 2015. Springer US, pp. 77–81. https://doi.org/10.1109/ICCNC.2015.7069319, 2015.

Najam, M., Younis, U., Rasool, R.U., 2015. Speculative parallel pattern matching using stride-k DFA for deep packet inspection. J. Netw. Comput. Appl. 54, 78–87. https://doi.org/10.1016/j.jnca.2015.04.013.

Oshima, S., Nakashima, T., Sueyoshi, T., 2010. Early DoS/DDoS detection method using short-term statistics. In: CISIS 2010 - the 4th International Conference on Complex, pp. 168–173. https://doi.org/10.1109/CISIS.2010.53. Intelligent and Software Intensive Systems.

Phan, T.V., Bao, N.K., Park, M., 2017. Distributed-SOM: a novel performance bottleneck handler for large-sized software-defined networks under flooding attacks. J. Netw. Comput. Appl. 91, 14–25. https://doi.org/10.1016/j.jnca.2017.04.016.

Phemius, K., Bouet, M., Leguay, J., 2014. DISCO: distributed SDN controllers in a multi-domain environment. In: IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World. https://doi.org/10.1109/NOMS.2014.6838273.

Piedrahita, A.F.M., Rueda, S., Mattos, D.M.F., Duarte, O.C.M.B., 2015. FlowFence: a denial of service defense system for software defined networking. In: Global Information Infrastructure and Networking Symposium. https://doi.org/10.1109/GIIS.2015.7347185. GIIS 2015 2015.

Qazi, Z.A., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., Yu, M., 2013. SIMPLE-fying middlebox policy enforcement using SDN. In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM - SIGCOMM '13, vol. 27. https://doi.org/10.1145/2486001.2486022.

Rodrigues Prete, Ligia, Shinoda, A.A., Schweitzer, C.M., de Oliveira, R.L.S., 2014. Simulation in an SDN network scenario using the POX Controller. In: IEEE Colombian Conference on Communications and Computing (COLCOM). IEEE, pp. 1–6. https://doi.org/10.1109/ColComCon.2014.6860403, 2014.

Sahay, R., Blanc, G., Zhang, Z., Debar, H., 2017. ArOMA: an SDN based autonomic DDoS mitigation framework. Comput. Secur. 70, 482–499. https://doi.org/10.1016/j.cose.2017.07.008.

Saied, A., Overill, R.E., Radzik, T., 2016. Detection of known and unknown DDoS attacks using Artificial Neural Networks. Communications in Computer and Information Science 172, 385–393. https://doi.org/10.1007/978-3-319-07767-3_28.

Santanna, J.J., Van Rijswijk-Deij, R., Hofstede, R., Sperotto, A., Wierbosch, M., Granville, L.Z., et al., 2015. Booters - an analysis of DDoS-as-a-service attacks. In: Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM, pp. 243–251. https://doi.org/10.1109/INM.2015.7140298, 2015.

Shamsi, J., Brockmeyer, M., 2012. Predictable service overlay networks: predictability through adaptive monitoring and efficient overlay construction and management. J. Parallel Distrib. Comput. 72, 70–82.

Shamsi, J., SyncProbe, Brockmeyer M., 2007. Providing assurance of message latency through predictive monitoring of Internet paths. In: Proceedings of IEEE International Symposium on High Assurance Systems Engineering, pp. 187–196. https://doi.org/10.1109/HASE.2007.69.

Shang, F., Mao, L., Gong, W., 2017. Service-aware adaptive link load balancing mechanism for Software-Defined Networking. Future Gener. Comput. Syst. https://doi.org/10.1016/j.future.2017.08.015.

Shin, S., Porras, P., Yegneswaran, V., Fong, M., Gu, G., Tyson, M., et al., 2013a. Fresco: modular composable security services for software-defined networks, 2. Network and Distributed System Security.

Shin, S., Yegneswaran, V., Porras, P., Gu, G., 2013b. AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. In: ACM SIGSAC Conference on Computer & Communications Security (CCS 2013). https://doi.org/10.1145/2508859.2516684, 413–24.

Sookhak, M., Tang, H., He, Y., Yu, F.R., 2018. Security and Privacy of Smart Cities: A Survey, Research Issues and Challenges. IEEE Communications Surveys & Tutorials.

Suh, J., Choi, H., Yoon, W., You, T., Kwon, T.T., Choi, Y., 2010. Implementation of content-oriented networking architecture (CONA): a focus on DDoS countermeasure. 1st European NetFPGA Developers Workshop 1–5.

Team, M., 2012. Mininet: an instant virtual network on your laptop (or other PC). 2015-11-15]. Http://Mininet Org 2012.

Technol, I., 2015. Secure and dependable SDNs. Journal of Network and Computer Applications Journal.

Tirumala, A., Qin, F., Dugan, J., Ferguson, J., Gibbs, K., 2005. Iperf: the TCP/UDP bandwidth measurement tool http://Dast Nlanr Net/Projects.

Tootoonchian, A., Ganjali, Y., 2010. Hyperflow: a distributed control plane for openflow. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking. https://doi.org/10.1109/JPROC.2014.2371999, 3–3.

Vitunskaite, M., He, Y., Brandstetter, T., Janicke, H., 2019. Smart Cities and Cyber Security: Are We There yet? A Comparative Study on the Role of Standards, Third Party Risk Management and Security Ownership. Computers & Security.

Wang, W., Lu, Z., 2013. Cyber security in the smart grid: survey and challenges. Comput. Network. 57, 1344–1371. https://doi.org/10.1016/j.comnet.2012.12.017.

Wang, H., Xu, L., Gu, G., 2015a. FloodGuard: a DoS attack prevention extension in software-defined networks. In: Proceedings of the International Conference on Dependable Systems and Networks 2015. https://doi.org/10.1109/DSN.2015.27. Septe:239–50.

Wang, R., Jia, Z., Ju, L., 2015b. An entropy-based distributed DDoS detection mechanism in software-defined networking. In: Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015, vol. 1, pp. 310–317. https://doi.org/10.1109/Trustcom.2015.389.

Xia, Wenfeng, Wen, Yonggang, Foh, Chuan Heng, Niyato, D., Xie, Haiyong, 2014. A survey on software-defined networking. IEEE Communications Surveys & Tutorials 17. https://doi.org/10.1109/COMST.2014.2330903, 1–1.

Yadav, V.K., Trivedi, M.C., Mehtre, B.M., 2016. DDA: an approach to handle DDoS (Ping flood) attack. In: Advances in Intelligent Systems and Computing, vol. 408. Springer, pp. 11–23. https://doi.org/10.1007/978-981-10-0129-1_2.

Yan, Q., Gong, Q., Yu, F.R., 2017. Effective software-defined networking controller scheduling method to mitigate DDoS attacks. Electron. Lett. 53, 469–471. https://doi.org/10.1049/el.2016.2234.

Yeganeh, S.H., Tootoonchian, A., Ganjali, Y., 2013. On scalability of software-defined networking. IEEE Commun. Mag. 51, 136–141. https://doi.org/10.1109/MCOM.2013.6461198.

Yu, S., Zhou, W., Jia, W., Guo, S., Xiang, Y., Tang, F., 2012. Discriminating DDoS attacks from flash crowds using flow correlation coefficient. IEEE Trans. Parallel Distrib. Syst. 23, 1073–1080. https://doi.org/10.1109/TPDS.2011.262.

Yu, J., Wang, Y., Pei, K., Zhang, S., Li, J., 2016. A load balancing mechanism for multiple SDN controllers based on load informing strategy. In: 18th Asia-Pacific Network Operations and Management Symposium, APNOMS 2016: Management of Softwarized Infrastructure - Proceedings. https://doi.org/10.1109/APNOMS.2016.7737283.

Yu, S., Zhou, W., Guo, S., Guo, M., 2016. A feasible IP traceback framework through dynamic deterministic packet marking. IEEE Trans. Comput. 65, 1418–1427. https://doi.org/10.1109/TC.2015.2439287.

Yu Hunag, C., Min Chi, T., Yao Ting, C., Yu Chieh, C., Yan Ren, C., 2010. A novel design for future on-demand service and security. In: International Conference on Communication Technology Proceedings, ICCT. https://doi.org/10.1109/ICCT.2010.5689156, 385–8.

Zaalouk, A., Khondoker, R., Marx, R., Bayarou, K., 2014. OrchSec: an orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions. IEEE/IFIP NOMS. In: IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World 2014. https://doi.org/10.1109/NOMS.2014.6838409.

Zhou, Y., Zhu, M., Xiao, L., Ruan, L., Duan, W., Li, D., et al., 2015. A load balancing strategy of SDN controller based on distributed decision. In: Proceedings - 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2014, pp. 851–856. https://doi.org/10.1109/TrustCom.2014.112.

Jawwad A. Shamsi completed his PhD. in Computer Science from Wayne State University, MI, USA in 2009. He is currently a Professor and Director of National University of Computer and Emerging Sciences, Karachi campus. His research interest lies in Distributed Systems, Networks, Security, and High Performance Computing. His research has been funded by NVIDIA Research Center and HEC NRPU grants. He has over 50 research publications in reputable journals and conferences.

Narmeen Zakaria Bawany completed her PhD in Computer Science from National University of Computer and Emerging Sciences, Karachi. She is currently working as an Associate Professor and Head of Computer Science and Software Engineering department at Jinnah University for Women, Karachi. She has over 15 years of teaching experience at graduate and under graduate level. She has supervised many under graduate projects and had also received funding from Ignite (National Technology Fund, Pakistan) for her projects. Her research areas include Human Computer Interaction, Semantic Web, Cyber security and Software Defined Networking.