IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 25, 2019, accepted August 23, 2019, date of publication September 6, 2019, date of current version October 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939780

On the Security of SDN: A Completed Secure and Scalable Framework Using the Software-Defined Perimeter

AHMED SALLAM[1,3], AHMED REFAEY[1,2], AND ABDALLAH SHAMI[1]
[1]Department of Electrical and Computer Engineering, Western University, London, ON N6A 3K7, Canada
[2]Manhattan College, Riverdale, New York, NY 10471, USA
[3]Department of Computer Science, Suez Canal University, Ismailia 41522, Egypt

Corresponding author: Ahmed Refaey (ahmed.hussein@manhattan.edu)

ABSTRACT The widespread adoption and evolution of Software Defined Networking (SDN) have enabled the service providers to successfully simplify network management. Along with the traffic explosion, there is decreasing CAPEX and OPEX as well as an increase in the average revenue per user. However, this wide adoption of SDNs is posing real challenges and concerns in terms of security aspects. The main challenges are how to provide proper authentication, access control, data privacy, and data integrity among others for the API-driven orchestration of network routing. Herein, the Software Defined Perimeter (SDP) is proposed as a framework to provide an orchestration of connections. The expectation is a framework that restricts network access and connections between objects on the SDN-enabled network infrastructures. There are several potential benefits as a result of the integration between SDP systems and SDNs. In particular, it provides a completely scalable and managed security solution. Consequently, it leads to flexible deployment that can be tailored to fit the need of any generic network security perimeter. The proposed Integrated frameworks are examined through virtualized network testbeds. The testing results demonstrate that the proposed framework is malleable to both port scanning (PS) attack and Denial of Service (DoS) bandwidth attack. In addition, it clarifies some interesting potential integration points between the SDP systems and SDNs to further research in this area.

INDEX TERMS SDP, SDN, DoS attack, security, network virtualization.

I. INTRODUCTION

Cisco predicts that by 2022 mobile devices will account for 79% of Internet traffic in comparison to the 65% share as of 2017 [1]. When coupled with the statistic that global IP traffic is expected to triple from 2017 to 2022 [1], the need for infrastructure innovation to keep up with the ever-changing landscape of users, resources, services, and applications is a must. In fact, traditional hardware-based networks work inadequately operate for consistently changing computing and storage needs in campus environments, data centers, and carrier/service provider environments. As a consequence, the Software-defined networking (SDN) gained significant traction as it fulfilled such situations, where numerous characteristics demand a more flexible and dynamic approach [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Tawfik Al-Hadhrami.

Despite the existing advantages of SDN security [3], [4], there are still aspects of security that remain to be addressed. For example, the centralized controller and flow-table limitation in network devices make the SDN-based network more vulnerable to Denial-of-Service (DoS) types of attacks [5]. In addition, the open programmability of the network introduces trust concerns between the network elements, which make threats by entering the network, and remain invisible and uninspected. Furthermore, compromised security of the controller or lapses in its datapath communication can render the whole network compromised or at least leave it vulnerable to illegitimate access and usage of network resources.

A number of industry-centered groups have been launched, as of late, to discuss the impending security challenges in SDN and their solutions. Meanwhile, researchers have already presented solutions to some SDN security challenges. These solutions range from controller replication schemes to policy conflict resolution and authentication mechanisms.

However, when the extent of the issues are compared to the existing and proposed solutions placed on them, it is clear that without a significant framework, equivalent to the SDN but focused on security, it is unlikely that SDN will succeed beyond the private datacenter or single organization deployments seen today.

An equivalent framework is the Software-defined Perimeter (SDP). The SDP is an independent framework, made popular its use by multiple organizations within the Department of Defense (DoD) and Intelligence communities (IC). In 2014 and emendate in 2018, the Cloud Security Alliance (CSA) outlined the initial protocol for the Software Defined Perimeter specifications [6] and in 2016 Waverley labs developed opensource SDP modules. Similar to the software-defined networking, the SDP has emerged as a new concept to replace physical security appliances with logical components that can operate under the control of the application owner. Although there is a conceptual similarity between these two independent frameworks, no such integration has been introduced and evaluated. Therefore, contributions of this paper can be summarized as follows:

- Propose an integrated SDP-SDN architecture to provide a better security networking platform and ensure seamless integration between the two paradigms.
- Build a virtualized network testbed to introduce and evaluate the aforementioned architecture.

The rest of this paper is organized as follows: In Section 2, previous works conducted on SDN security challenges and solutions are reviewed. In Section 3, SDP architecture is presented with an explanation of its functionality. In Section 4, an introduction to possible integration architectures with corresponding challenges. In Section 5, new integrated architecture is evaluated. Finally, the work and research provided are concluded in Section 6.

II. RELATED WORK

The large inflation in the world of networks after the advent of virtualization technology and electronic cloud led to the need of separating the control plane from the data/forwarding plane in a new model that can elastically expand or shrink with the dynamic change on the network without affecting the network overall performance. Usually, the SDN network can be divided into three main levels as shown in Fig. 1. The data plane (Southbound), the control plane, and the application plane (Northbound) [7], [8].

The Control plane is managed by an SDN controller. The basic functions of the controller include flow table management, link discovery, topology management, strategy making, storage management, and control data management. Consequentially, many applications and features can be added as needed.

On the other hand, The data plane consists of a set of forwarding devices which are commonly known as SDN switches, although they may not contain the basic function of layer 2 switches. The SDN switch can be found as software

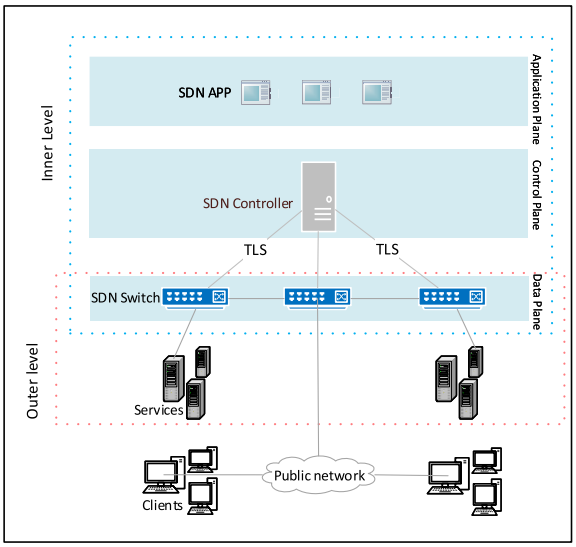

FIGURE 1. SDN architecture.

Port	Src MAC	Dst MAC	VLAN ID Priority	EtherType	Src IP	Dst IP	IP Proto	IP ToS bits	Src TCP/UDP port	Dst TCP/UDP port	Action	Counter
*	*	*	*	*	*	*	*	*	*	*	Port 1	125
*	C5:0A	*	*	*	*	10.10.30.5	*	*	45	45	Port 2	345
*	*	*	*	*	*	*	*	*	*	*	Drop	102
*	*	*	*	*	*	192.*	*	*	25	25	Local	210
*	*	*	*	*	*	*	*	*	*	*	Controller	3

FIGURE 2. SDN flow table.

such as OpenvSwitch [9] and can be found as hardware router or switch that supports one of the southbound protocols such as Cisco Catalyst 2960-S Series which supports Simple Network Management Protocol (SNMP) and HP ProCurve switches which supports OpenFlow. Moreover, hardware switches can be fully programmable such as Broadcom Maverick switch which supports OpenSwitch network operating system (OPX).

An SDN switch consists of one or more flow tables which perform packet lookups and forwarding (see Fig. 2), and a southbound channel to communicate with an external controller. Using this channel, the controller can add, update, and delete flow entries in flow tables, both reactively (in response to packets) each time the switch receives a packet with no matching rule in its flow table and proactively where the rules are loaded to the switch when the network starts. Each flow table in the switch contains a set of flow entries; each flow entry consists of match fields, counters, and a set of instructions to apply to match packets [10].

To understand the control flow between the controller and the SDN switch, consider a scenario of an OpenFlow switch which received a packet with no matching rule in its flow table. In this scenario, a client sends a service request to the switch, and then the switch starts the matching process at

TABLE 1. SDN security challenges and existing solutions.

Scope	Targets	Challenges	Existing solution	Type	Drawbacks
Outer level	- Services - Switches	- Flow tables memory limitation-Performance degradation - Vulnerability of Flooding attacks (e.g. DoS attack)	Security Middleboxes (e.g. firewall, IDS and anti-maleware)	HW	- Cannot integrate into virtualized environment - Cost
			Software-defined Security (e.g. vArmour and VMware vShield)	SW	- Poor performance against massive attacks - Cost
			Machine Learning classification techniques	SW	Exhaust computing resources and resulted in network overhead
Inner level	- Controller - Southbound/ Northbound Interfaces - SDN APPs	- Single point of failure (Controller compromises) - Network manipulation (Controller hijacking) - lack of Authorization & Authentication - lack of Encryption - Performance Degradation - Subject to Sniffing and Spoofing attacks	Encrypted channel	HD/SW	- Is not supported by all SDN switches and controllers - Don't encrypt all the data being transferred
			Access Control List (ACL)	SW	Difficult to manage and utilize

the first flow table and if no match was found the switch continues to next flow tables. The outcome depends on the configuration of the table-miss flow entry; For example, the packet may be dropped, or forwarded to the controller over the OpenFlow channel which is the case of this scenario. Next, the controller will add two flows to connect the new client to the service. One flow to set the client as a source and the service as a destination, and another flow to set the opposite direction.

Although SDN allows virtual networks provision on demand for both efficient data transport and fine-grained control services [11], [12], current security practices were not designed to match the complexity and challenges emergent from these software-defined infrastructures' integration [13]. Precisely, separation of control and data planes opens security challenges categorized into two levels. The first/outer level is to protect the servers and switches from malware that can sniff metadata and flooding attacks, which can result in whole security systems take-down. The second/inner level is to prevent malicious nodes from penetrating the controller and taking over the network. The security challenges and their existing solutions and drawbacks are summarized in Table 1.

Table 1 displays the security challenges in SDN network categorized as either outer or inner levels. A further discussion of these challenges and existing solutions are provided in the following subsections. In addition, further details are available in [14]–[17].

A. SDN OUTER LEVEL SECURITY

In the SDN systems' outer level, services encounter flooding attacks such as Denial of Service (DoS) and Distributed DoS (DDoS). The attack's target is to expose the dedicated server resources and/or to slow legitimate users causing a denial of service [18]. Consequently, SDN switches endure severe performance degradation depending on SDN-controller decisions made while being attacked. Notably, other factors affect switch performance, such as, flow tables, memory and forwarding rate limitations, however, these factors are beyond this work's scope.

Traditionally, detecting flooding attacks is achieved by installing hardware-based middleboxes deployed with dedicated security functions such as intrusion detection (IDS), firewalls, and anti-malware. In fact, these hardware-based equipment are incapable of detecting significant security activities inside modern software-based core networks in general, and in SDN-based networks, in particular [19]. Therefore, Software-Defined Security (SDSec) has been introduced [19]. The SDSec, a relatively new software-based approach, separates forwarding and processing planes from the security control plane following the same logical concept of SDN. This concept has been adopted by industries for distributed security in appliances by virtualizing security functions into ready-to-use Virtual Machines (VM). vArmour, VMware vShield and Catbird are a well-known example which implements a number of security features and attributes such as intrusion detection, anti-malware, and firewalls.

Machine Learning (ML) classification techniques were explored to implement a security mechanism to detect and prevent malicious traffic [15]. These techniques usually include three steps: classify data flows to determine malicious behavior and network attacks; modify flow tables; then check the computed flow rules. Indeed, adopting these techniques in SDN-based networks will contribute to increased

overhead (e.g. collecting and transmitting traffic statistics) which overloads OpenFlow switches. Also, controllers must process statistics and compute flow rules before applying to the switches [15].

In general, neither hardware nor software-based solutions can withstand massive attacks that scale up to overwhelm most traditional on-premises equipment and resources available in any virtualized environment. Fore example, between May and June 2018, 8.3 billion malicious login attempts were reported by Akamai, one of the world's largest distributed computing platforms that responsible for serving between 15% and 30% of all web traffic. These malicious login attempts using account takeover tools known as botnet can cause rapid destruction dute to the large volume of generated traffic that reach over 600 Gbps [14]. Many copanies would typically treat this like a DDos attack.

B. SDN INNER LEVEL SECURITY

In the inner level of SDN systems, the centralized controller presents a potential single point of failure that is vulnerable to network manipulation. Consequently, an intruder could compromise the SDN controller and/or one of the SDN applications, produce false network data, and initiate different attacks on the entire network [16]. To protect the controller, a classical security aware solution, such as the access control list (ACL), is utilized to define permissions applying to an object and its properties. A security strategy involves creating a whitelist and blacklist, which is an inconvenient burden during large network configuration changes [20]. To eliminate this burden, security policy frameworks were developed to automate security policy transitions [20]. However, automation consumes more resources, especially in a virtualized environment with numerous VMs. Furthermore, no standards exist to facilitate control and application planes, seen as SDN-Northbound interface. Therefore, third-party SDN applications induce several active and passive attacks (e.g. protocol spoofing, infiltrate the network, and sniff-modify-stop traffic) [21].

Well-Established security mechanisms, like "honeypot", are used to mitigate potential threats to SDN applications by forming high-interactions called "Honeynet" [17]. Some leading honeypot systms include Google Hack Honeypot (GHH), ''KFSensor'' a commercial Windows-based honeypot Intrusion Detection System (IDS) and ''Honeyd'' an open source software released under GNU license for Unix Operating Systems [17]. Generally, a honeypot consists of data that appears to be a legitimate content of the site and that seems to contain valuable information for intruders but is monitored to track malicious behavior [22]. For complex infrastructure that hosts a variety of services such as SDN networks, one dedicated machine must be maintained for each honeypot, which can be exorbitantly expensive. This form of high-interaction honeypots is known as Honeynet.

Additional security-aware solutions are implemented to add more power to the controller such as the access control list (ACL) which define the permissions that apply to an object and its properties. This task is further compounded when we consider that network operators utilize whitelists and blacklists as part of their security strategy for no less than 18,000 network configuration changes [23]. To overcome this burden, many security policy frameworks were developed to automate the security policy transitions [20]. However, the side effect of this is that we are adding more burden over the controller which is already overwhelmed especially in a virtualized environment with numerous number of VMs.

Variously, data planes seen as SDN-Southbound interface, lack encryption. This gives intruders opportunities to sniff, capture, and analyze network traffic by allowing the eavesdropping of critical flow information within the SDN-based network. Traditionally, sniffing activity is prevented by Transport Layer Security (TLS) and/or Secure Socket Layer protocol (SSL) [24]. These encryption methods help protect connections between the SDN controller and switches. For example, OpenFlow protocol specification recommends using TLS connections between switches and the controller, however, some SDN switches and controllers do not support encryption, specifically the older versions [3], [25].

III. SOFTWARE-DEFINED PERIMETERS FRAMEWORK

As a matter of fact, perimeters security is an ancient method used by old empires to create a secure perimeter through walls and barriers to maintain areas of power and property away from intruders. In networking, this method was imitated by creating a boundary between the private side of a network and the public (usually provider-managed side of a network). Previously, network perimeters had been applied by installing hardware-based middleboxes deployed with dedicated security functions such as firewalls and IDS. Today, with the new Software-Defined paradigm, where resources are shared, what does the network really look like? It is clear that a network perimeter is an outdated concept for several reasons. First, internal users are not simply connecting from inside the building. They can connect anywhere using mobile devices. Another vital reason is, local data and private applications are no longer on local servers. Therefore, SDP emerged as a new concept to go along with this new paradigm. It is worth mentioning that, SDP market now is competitive and some of the top players in the SDP market such as Cisco Systems Inc., Symantec Corporation, and Intel Corporation. Accordingly, the Transparency Market Research estimates the SDP market's valuation to increase from US$1,129.5 mn in 2016 to US$12,247.9 mn by 2025 [26].

A. INTRODUCTION

Although there is a conceptual similarity between SDN and SDP, both are independent-standalone solutions. Precisely, SDN is the notation to control network behavior by emphasizing the software application instead of the network infrastructure. However, SDP is a completely different notation to secure the application network infrastructure based on a need-to-know model, in which device identity is verified before

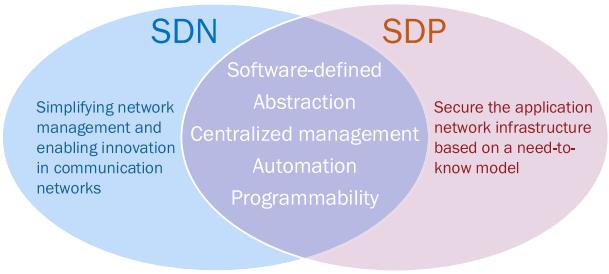

FIGURE 3. Commons between SDN and SDP.

granting access [27]. The commons and differences between the aforementioned frameworks are shown in Fig. 3.

Typically, the SDP paradigm consists of three main components:

- SDP Initiating Host (IH),
- SDP Accepting host (AH),
- SDP controller (CTRL).

These components are used to create secure perimeters among legitimate clients and available services in the network. A simple SDP scenario would include three machines; the first machine represents a legitimate client provided with the IH module and trying to access a server behind a gateway in a private network. The second machine represents the gateway provided with an AH module and a firewall that has a drop-all policy established for all traffic. The third machine represents the SDP controller provided with the CTRL module to manage the SDP authentication process.

During the SDP installation process, the network administrator should access the CTRL module to identify the legitimate clients and define services which they have access to in the CTRL database {MySQL in this scenario. Moreover, the administrator has to create credential keys and certificates and distribute them among the IH and AH components to authenticate their access to the CTRL.

B. SDP CONNECTION OVERVIEW

For a legitimate client to access a service, first, the IH module already installed in its side (also known as SPAclient) sends a valid Single Packet Authorization (SPA) packet (encrypted, non-replayed, with an HMAC SHA-256) [28]. When the CTRL authenticates the SPA packet, it will message the AH at the gateway to configure the proper rules automatically in the firewall for the client for a defined period to access the services. Despite the fact that the gateway's firewall is enabled, the AH continues to receive the message, authenticate the client and establish a Mutual Transport Layer Security (mTLS) connection between the client and gateway. After the configurable timeout, the rule to accept the incoming connection will be deleted but the connection remains open by using a tracking mechanism provided by the firewall [28]. Notably, the AH has to follow this procedure to initiate the connection to the controller. Fig. 4 summarizes the Client-Server connection setup timeline with SDP.

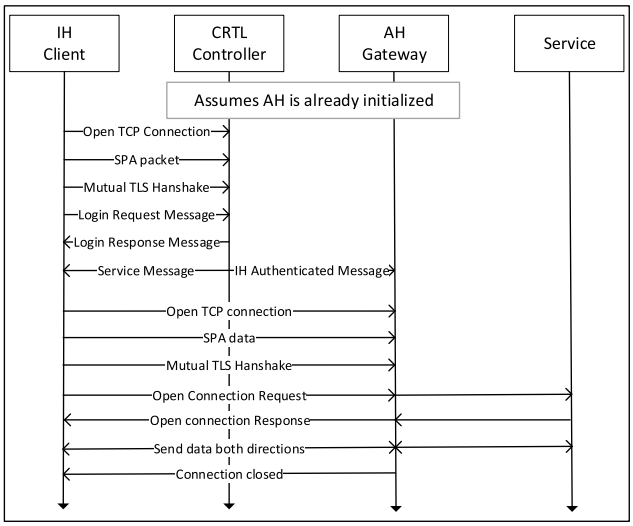

FIGURE 4. Client-Server connection setup timeline with SDP.

According to the CSA standards, the legitimate client has direct access to the SDP controller, this assumption was set to illustrate how the authentication process is done [6]. However, this configuration exposes the SDP controller to direct attacks that can lead to critical risk where an intruder can take over the whole network. This problem can be tackled by hiding the controller behind the gateway in a similar manner to the services.

C. SDP PROTECTION

The SDP workflow through several layers of security gives maximum protection to the systems while patching most vulnerabilities found in the legacy security systems. Firstly, the gateway\textquoteright s firewall contains a static drop-all policy allowing SDP to effectively repel flooding and PS attacks. Secondly, the SPA mitigates these attacks by allowing the server to discard the DoS attempt before entering the TCP handshake [6]. Thirdly, the connection between all hosts (IH and AH) must use TLS or Internet Key Exchange (IKE) with mutual authentication to validate the client as a legitimate member of the SDP prior to furthering device validation and/or user authentication.

All of these layers are capable of preventing network manipulation attacks, MITM attacks, and traffic sniffing attacks. In fact, the CSA SDP Hackathon challenged hackers to attack a server defended by a SDP. Of the billions of packets fired at the server, not one attacker penetrated even the first layer of security [29].

IV. THE PROPOSED SDP-SDN INTEGRATED ARCHITECTURE

In this section, the SDN networks ability to provide protection through SDP is presented. The following subsections demonstrate the proposed architecture integration between SDP and SDN main components. An explanation of various client-server request scenarios is also provided.

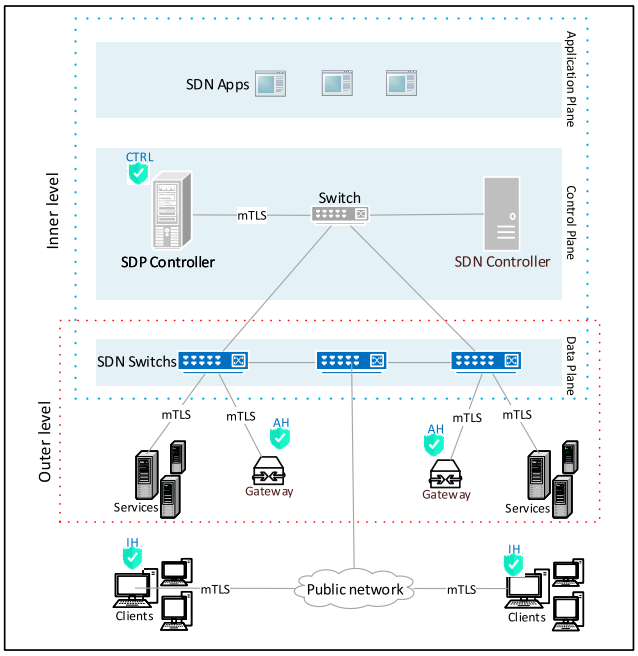

FIGURE 5. Integrating SDP with SDN.

A. ARCHITECTURE DESCRIPTION

The goal is to protect a set of services connected to an SDN network by embedding the SDP components into the SDN environment. To complete this task, each legitimate client runs an IH module. Additionally, a separate gateway machine connects to each SDN switch. Each gateway runs as an AH module to manipulate a set of servers/services connected to the switch as depicted in Fig. 5. Finally, representing the SDP controller, another machine running the CTRL module is added to the network. To clarify, the SDP controller was placed on a separate machine.

Refereeing to the CSA standards, there are several ways to implement the SDP-based network depending on the targeted application. These implementation options can be summarized as follows:

- **Client-to-Gateway:** in this implementation, the service/server is hidden behind a gateway provided with a running AH module.
- **Client-to-Server:** in this implementation, the AH module can be installed directly into the server, such that the server can act as a gateway and a service at the same time.

Alternative implementation options, such as Server-to-Server and Client-to-Server-to-Client are out of the scope of this work due to their irrelevancy to SDN.

B. THE GATEWAY PLACEMENT

Choosing a suitable SDP implementation to integrate with SDN is a trade-off between security and performance. On the one hand, the Client-to-Server implementation is harmonized with SDN because installing the AH module in each server reduces the routing path to access this server instead of going through extra nodes in the network, especially in a dynamic environment with a huge number of VMs [30]. However, this implementation is considered a poor security practice. Precisely, it exposes the server to direct attacks, such that the SDP gateway only acts as a regular firewall with an advanced protocol. On the other hand, the Client-to-Gateway implementation seems to increase delay by placing additional gateway in the middle to access the servers. However, it provides a more secure environment by hiding these servers and reducing the risk of taking them down (see Fig. 5). In fact, Client-to-Gateway implementation does not completely counteract the SDN network performance, however, it can reduce the flow control delay between the SDN's controller and switches. To understand this point, assume that there exists two servers S1 and S2 behind the gateway G1 and both connected to the same SDN switch SW1. Suppose that a legitimate client C1 wants to access S1. In this case, the SDN controller will add two flows to SW1 to provide a bi-directional link between C1 and G1. Now, suppose C1 wants to access S2, in this case, SW1 will do nothing because it already has predefined flows between C1 and G1.

When using Client-to-Gateway the open research question to consider is, how many gateways would provide maximum security while reducing network overhead in SDN platform? To answer this question, assume that only one gateway hides all the servers. Although this scenario will reduce the flow control delay to its minimum, it will degrade the overall network throughput by adding more burden on this gateway and put it at risk of network bottlenecks. Now, assume a completely different scenario by assigning a gateway to hide each server. This scenario provides a better security solution and keeps the network performance untouched; however, it will consume more resources and increase overall cost.

The service provider decides the appropriate number of gateways based on a cost-to-benefit analysis and the desired performance metric (throughput, delay). Herein, one gateway was assigned to hide all services connected to the same SDN switch. This strategy can provide excepted protection and avoids losing more resources at extra cost.

C. FLOW CONTROL

Two scenarios considered here were based on the placement of the SDP Controller with respect to the IH.

Scenario 1: In this scenario, the IH has direct access to the SDP Controller, and the AH is already initialized. Thus, for a new legitimate Client (A) to access an authorized service (F), A should complete two communication sequences:

1) Authentication: The IH module on A will initialize an authentication request to the SDP Controller (D). Because (A↔D) are two new flows to the SDN Switch, the Switch will send a packet-in to the SDN Controller (C) to reply with the correct flows from A to D and back from D to A as shown in Fig. 6. Next, the SDP CTRL module on D will check the required

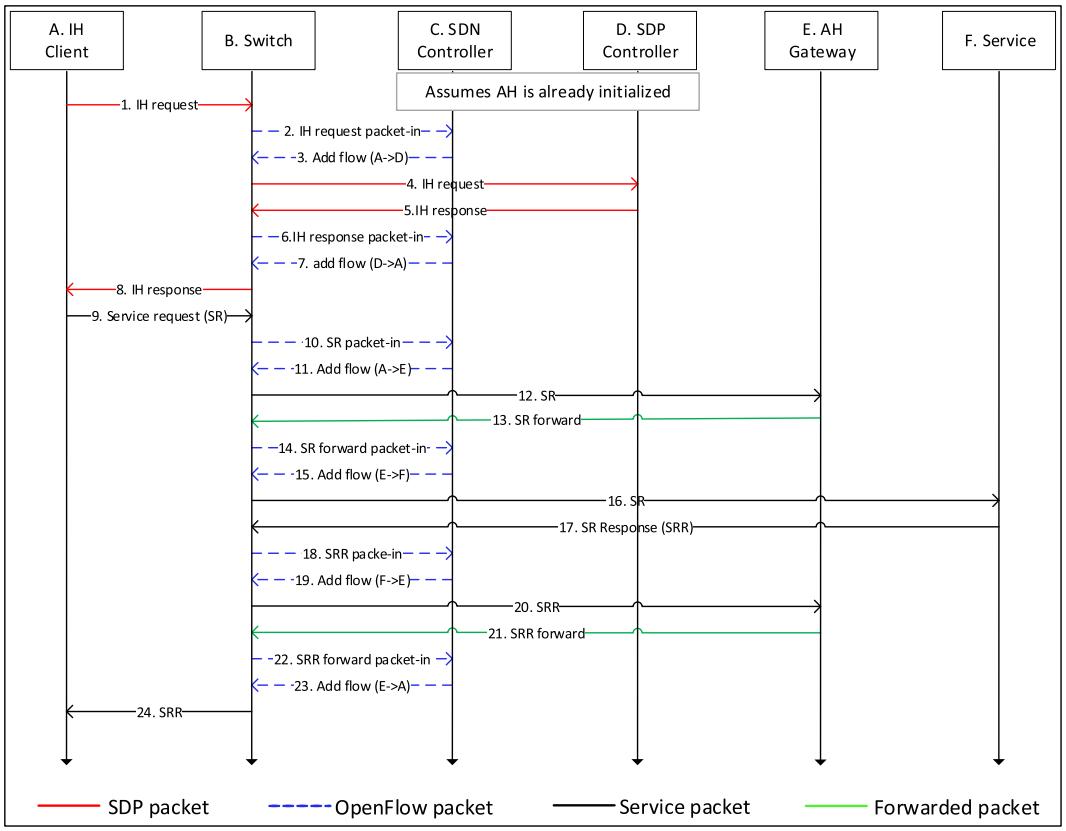

FIGURE 6. **Message sequence in the new platform - Senario 1.**

credential and authenticates A to access F through the gateway (E). If A passed the authentication process, new forwarding rules added to the firewall (Iptables in this scenario) at the gateway (E) to allow communication between A and F.

2) Service request: After A has been authenticated to access F through E. Now, A can send the desired service request to a running service on F through E as shown with black arrows in Fig. 6. Again, because (A↔E) are new flows to the Switch, the Switch will send a packet-in to C one more time to get the correct flows from A to E and back from E to A. When E receives the service request, it will forward it to F and forward back the response to A using the rules created in step 1 as shown with green arrows in Fig. 6.

Scenario 2: As mentioned before, it's a bad security practice to give a client direct access to the SDP controller. To tackle this problem, in this scenario, the SDP controller is placed behind the gateway as shown in Fig. 7. In other words, any communication between the Client and the SDP Controller will go through the gateway. This has two advantages. Firstly, to protect the SDP Controller. Secondly, to reduce the flow control delay time as there will be no need to define new flow rules between each client and the SDP controller.

It's worth mentioning that, the flow control requests (packet-in) shown with dashed lines in Fig. 6 are one-time requests to create the proper flows between the clients, gateway, and services. In other words, if there exists client C that already gained access to services F1 through the gateway, there will be no flow control delay if C wants to access another service F2 through the same gateway.

V. TESTBED AND PERFORMANCE EVALUATION

To evaluate the performance of the proposed platform, three evaluation metrics are considered, the flow control delay time, the SDP connection setup time and the network throughput. These metrics were compared with and without SDP installed. Additionally, two types of attacks were initiated to test the performance of SDP protection, namely, a DoS, and a PS attacks. The DoS attack was chosen as it represents a threat to the services availability while the PS attack represents a threat to data privacy. Therefore, these two attacks were considered as they represent two threats the SDN networks expected to experience.

A. TESTBED ENVIRONMENT

The testbed consists of five VMs running Linux Ubuntu 16.04 hosted by a physical machine running Ubuntu 18.04, and VirtualBox 5.2. The implementation can be divided into two parts, the SDN part, and the SDP part. The implementation of the SDN part has been done using OpenDaylight (ODL) controller and Open vSwitch (OVS) 2.5.6. OVS

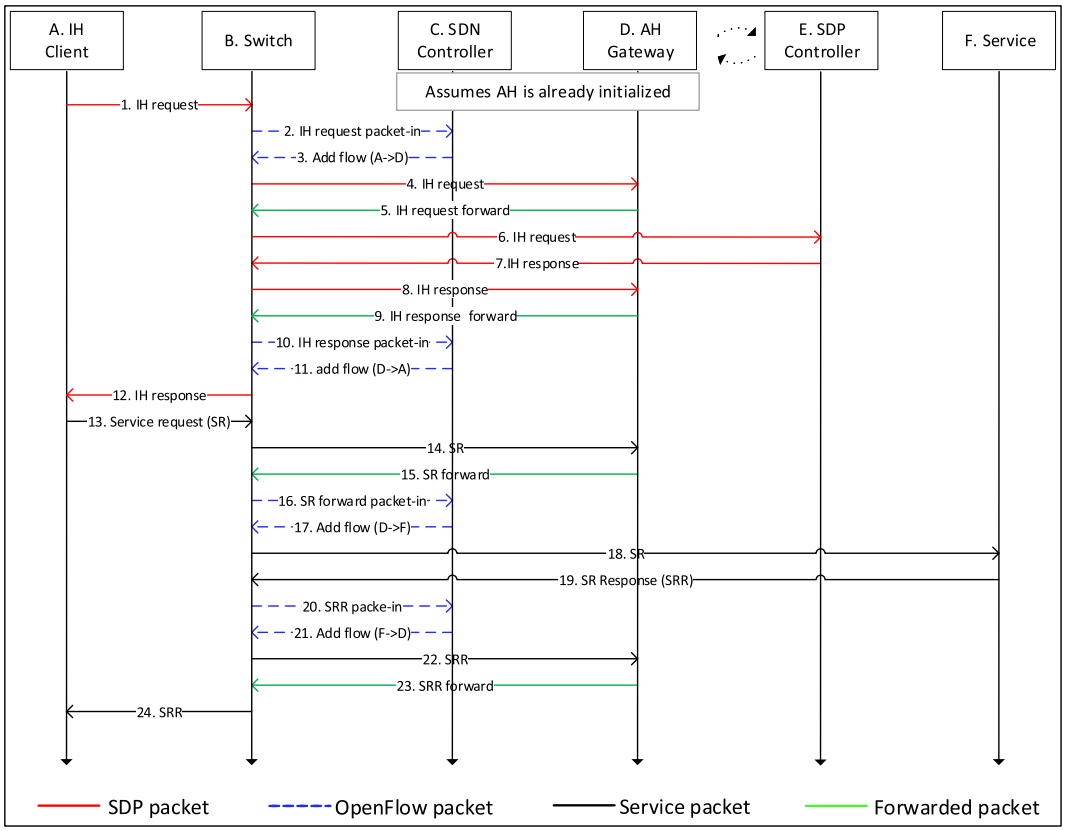

FIGURE 7. Message sequence in the new platform - Senario 2.

TABLE 2. Testbed setup.

Machine	Software	Specifications
Host	- Linux Ubuntu 18.04 - VirtualBox 5.2, - Open vSwitch 2.5.6	- 4.20GHz Intel® Core™ i7-8650U, 8MB Cache - 32 GB DDR4 - 2GB GDDR5 NVIDIA Quadro P500 - 1.0 Gbps Full Duplex Ethernet card
VM1 (SDP Controller)	- Linux Ubuntu 16.04 - Waverely SDPcontroller module	- 1 vProcessor - 1 GB RAM - 2 NICs
VM2 (Gateway)	- Linux Ubuntu 16.04 - Waverely fwknop module (AH configuration)	- 1 vProcessor - 2 GB RAM - 3 NICs
VM3 (Legitimate Client)	- Linux Ubuntu 16.04 - Waverely fwknop module (IH configuration)	- 1 vProcessor - 1 GB RAM - 2 NICs
VM4 (Server)	- Linux Ubuntu 16.04	- 1 vProcessor - 1 GB RAM - 2 NICs
VM5 (SDN Controller)	- Linux Ubuntu 16.04 - OpenDaylight Boron	- 1 vProcessor - 2 GB RAM - 2 NICs

was installed within the host, while ODL was installed in a dedicated VM. The SDPcontroller module was installed on a VM to represent the SDP controller and the fwknop module was installed on another VM to represent the SDP IH on a legitimate client with the help of MySQL 5.7 and OpenSSL 1.1. Finally, the fwknop module was installed again with different settings on a separate VM to represent the SDP AH and act as the gateway with the help of iptables 1.8 for routing. One more VM was created to act as a service. A detailed specification of the testbed setup is showen in Table 2.

B. ATTACKS SETUP

Two types of attacks where launched using hping3. hping3 is a command-line oriented TCP/IP packet assembler/analyzer.

- **DoS attack:** This attack was applied via spoofed broadcast of TCP requests with SYN flag.

- **Port Scanning attack:** This attack was applied with SYN flag.

C. EXPERIMENTS FORMULATION

The SDP connection setup was tested by calculating the required time to initiate each SDP component in a regular network with L2Switch and in an SDN network with OpenFlow enabled switch. The result shows that there was no significant difference between the startup time of the SDP components in both environments. That is, the accepting host authentication time in the regular network takes 4.182067 s while it takes 4.194367 s in SDN network, and the the Initiating Host authentication time takes 2.068458 s while it takes 2.067069 s in the SDN network.

Secondly, the network throughput was reported by transferring 250 MB of raw data between the legitimate client and the service using netcat utility version 1.10. The traffic was captured on the gateway from the client side and the server side for two scenarios, when the SDP platform is disabled and when the SDP platform is enabled (see Fig. 8 and Fig. 9).

Finally, the flow control delay time between the Packet-in message sent by the OVS switch and the Flow_MOD replied back by the ODL controller was calculate as 45 ms.

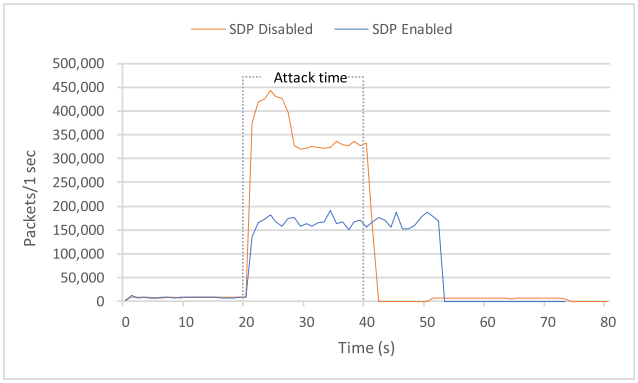

FIGURE 8. Network traffic at the gateway - client side.

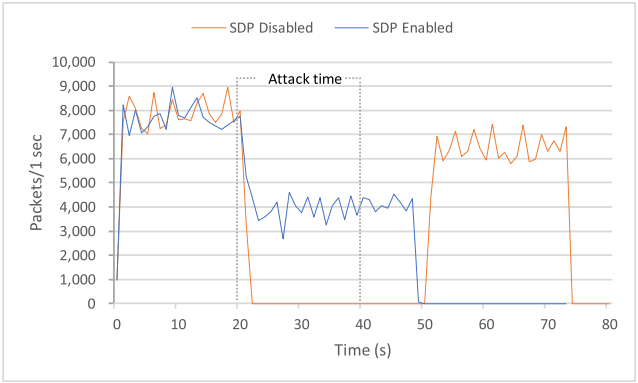

FIGURE 9. Network traffic at the server.

The port scanning attack was launched by sending a SYN flag request to ten consequent destination ports starting from port 50. Normally, if they are open then they will reply on the source port. The result was captured in Fig. 10.

```
root@attacker1:~# hping3 -S 10.10.30.101 -p ++50 -c 10
HPING 10.10.30.101 (enp0s8 10.10.30.101): S set, 40 headers + 0 data bytes

--- 10.10.30.101 hping statistic ---
10 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@attacker1:~#
```

FIGURE 10. Port scanning attack with SYN flag.

D. RESULT ANALYSIS

From Subsection V-C, we can notice that there is insignificance delay by running the SDP components in SDN network compared to the regular network due to a one-time flow control delay time required to find the correct flow between the gateway/client and the SDP controller. The reason behind this is that when a machine is trying to talk to another machine through an SDN switch (OVS in this environment) the switch would search first in its flow tables to find a route to the destination, if the switch cannot find the correct flow it will start a flow control sequence by sending a Packet-in message to the controller.

On the other hand, Fig. 8 and Fig. 9 show how did the SDP platform mitigate the DoS attack and retain the legitimate client connected. In this figure, the number of packets was captured at the gateway's the public network interface and the Server's network interface, and then a log function was applied to relax the different scale of both traffic.

Considering the traffic at the Gateway's public network interface shown in Fig. 8, from the second 0 to 19 the traffic represents the authorized packets of the legitimate client. At the second 20, a SYN flood attack was launched, this explains the sudden peaks in the traffic which reflects the effect of the attack. Without SDP platform, the traffic represents the attack's SYN packets to request a flood of new connections in addition to the ACK packets sent back to reply to the attacker. However, With SDP platform, this traffic represents the attack's SYN packets only, which clarify the difference between the number of packets in each line. In the latter case, the SYN packets are simply dropped and never replied back because the AH module initiates the firewall's drop policy at the gateway. This setting tells the firewall to allow only TCP packets combined with an authorized SPA packet. At second 40, the flood attack was stopped however its effect continues for a few seconds later based on the Ethernet interface buffer size and contents. This happens due to the massive number of packets received and buffered at the interface to be processed.

Considering the traffic at the server's network interface shown in Fig. 9, which represents the client's data packets. Starting from second 20 to second 40, without SDP, the client traffic was taken down completely by the attack and recovered back again after stopping the attack at second 40. Contrarily, the SDP retained 75% of the legitimate traffic

and the transmission of the data was successfully completed at second 50.

Fig. 10 shows that the SDP-SDN platform was able to completely block the port scanning SYN request due to the firewall drop policy.

VI. DISCUSSION

Although the proposed architecture can protect the SDN outer level, more challenges still exist to secure the inner level, mainly the SDN Controller. Generally, the SDN Controller is hidden by default and invincible against flooding attacks because of the SDN architecture. In other words, it's not possible to directly attack the SDN Controller unless metadata was sniffed which is not possible especially in the new proposed architecture due to the TLS connection. Moreover, there are several simple solutions that can be traded such as managing a firewall at the controller to block all inbound traffic except the SDN switches.

From a different perspective, there is still a chance for indirect attacks to take over the controller. For example, a DoS attack can be established by requesting anonymous servers randomly through one the SDN switches connected to the controller. Although these requests will end up in a null route or blocked by a firewall, it still can overwhelm the SDN Controller with Packet-In requests, where the controller compelled to provide the switch with a proper reply. Fortunately, this special DoS attacks can't be established from a public network/internet and requires direct physical access to the SDN switch which is managed by the targeted Controller.

VII. CONCLUSION AND FUTURE RESEARCH OPPORTUNITIES

This paper illustrated the potential of SDN networking security architecture by integrating the SDP framework with SDN as a solution to security challenges threatening different levels of the network. First, we briefly summarized and discussed challenges facing the SDN network. Then, we adopted a client-gateway SDP architecture to propose an improved and secured SDN network. Furthermore, performance was evaluated by analyzing network throughput and connection setup time under two types of network attacks, namely DoS and PS attack.

The experiments' result proved that by integrating the SDP and SDN frameworks, it is possible to block PS and flooding attacks. Meanwhile, mitigating its effect on the target resources to retain 75\% of the network throughput without interruptions or losing the connection. These results reveal the promising potential of the proposed architecture to provide a solution that is homogeneous with the SDN system and accommodates the networks' scale in a virtualized environment. Furthermore, it would reduce the Operating Expense (OPEX) and Capital Expense (CAPEX) of enterprises.

ACKNOWLEGMENT

The authors would like to thank Ms. Juanita Koilpillai from Waverley Labs for her valuable support.

REFERENCES

[1] Cisco. (2018). *White Paper: Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf

[2] K. Alhazmi, A. Shami, and A. Refaey, "Optimized provisioning of SDN-enabled virtual networks in geo-distributed cloud computing datacenters," *J. Commun. Netw.*, vol. 19, no. 4, pp. 402–415, Aug. 2017.

[3] S. Scott-Hayward, G. O'Callaghan, "SDN security: A survey," in *Proc. SDN4FNS Workshop Softw. Defined Netw. Future Netw. Services*, Nov. 2013, pp. 1–7.

[4] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 7, pp. 1838–1853, Jul. 2018.

[5] V. Yazici, M. O. Sunay, and A. O. Ercan, "Controlling a software-defined network via distributed controllers," *CoRR*, vol. 90, no. 11580, pp. 6–11, Jan. 2014. [Online]. Available: https://arxiv.org/abs/1401.7651

[6] A. Moubayed, A. Refaey, and A. Shami, "Software-defined perimeter (SDP): State of the art secure solution for modern networks," *IEEE Network*, vol. 33, no. 5, pp. 226–233, Sep./Oct. 2019.

[7] M. C. Dacier, H. König, R. Cwalinski, F. Kargl, and S. Dietrich, "Security challenges and opportunities of software-defined networking," *IEEE Security Privacy*, vol. 15, no. 2, pp. 96–100, Apr. 2017. doi: 10.1109/MSP.2017.46.

[8] V. Varadharajan, K. Karmakar, U. Tupakula, and M. Hitchens, "A policy-based security architecture for software-defined networks," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 897–912, Apr. 2019.

[9] M. Casado. (2018). *List of OpenFlow Software Projects (That I Know Of)*. [Online]. Available: http://yuba.stanford.edu/~casado/of-sw.html

[10] P. Goransson and C. Black, ""How SDN works," in *Software Defined Networks*, P. Goransson and C. Black, Eds. Boston, MA, USA: Morgan Kaufmann, 2014, ch. 4, pp. 59–79. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124166752000048

[11] K. Kalkan and S. Zeadally, "Securing Internet of Things with software defined networking," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 186–192, Sep. 2018.

[12] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with SDN: A feasibility study," *Comput. Netw.*, vol. 85, pp. 19–35, Jul. 2015. doi: 10.1016/j.comnet.2015.05.005.

[13] R. Khondoker and N. Function, *SDN NFV Security*, vol. 30. New York, NY, USA: Springer, 2018. [Online]. Available: http://link.springer.com/10.1007/978-3-319-71761-6

[14] E. Shuster, R. Shen, M. McKeay, and A. Fakhreddine, "State of the Internet," *Environment*, vol. 4, no. 4, p. 18, 2018. [Online]. Available: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/soti-2018-credential-stuffing-attacks-executive-summary.pdf

[15] B. J. Van Asten, "Increasing robustness of software-defined networks," M.S. thesis, Dept. Fac. Elect. Eng., Math. Comput. Sci., Delft Univ. Technol., Delft, The Netherlands, 2014.

[16] M. McBride, M. Cohn, S. Deshpande, M. Kaushik, M. Mathews, and S. Nathan, "SDN security considerations in the data center," ONF Solution Brief, Menlo Park, CA, USA, 2013.

[17] OmniSecu. (2018). *Leading Honeypot Products*. [Online]. Available: http://www.omnisecu.com/security/infrastructure-and-email-security/leading-honeypot-products.php

[18] A. F. T. Ali, R. Gziva, S. Jouet, and D. Pezaros, "SDNFV-based DDoS detection and remediation in multi-tenant, virtualised infrastructures," in *Guide to Security SDN NFV*. Cambridge, MA, USA: Springer, 2017, ch. 7.

[19] Y. Jararweh, M. Al-Ayyoub, E. Benkhelifa, M. Vouk, and A. Rindos, "Software defined cloud: Survey, system and evaluation," *Future Gener. Comput. Syst.*, vol. 58, pp. 56–74, May 2016.

[20] J. H. Cox, R. J. Clark, and H. L. Owen, "Security policy transition framework for software defined networks," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2017, pp. 56–61.

[21] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.

[22] E. Cole and S. Northcutt. (2018). *Security Laboratory Honeypots: A Security Manager's Guide to Honeypots Honeypot Liabilities*. [Online]. Available: https://www.sans.edu/cyber-research/security-laboratory/article/honeypots-guide

[23] H. Kim, J. Reich, A. Gupta, M. Shahbaz, N. Feamster, R. Clark, and I. Nsdi, "Kinetic: Verifiable dynamic network control," in *Proc. 12th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2015, pp. 59–72.

[24] K. Bhargavan, B. Blanchet, and N. Kobeissi, "Verified models and reference implementations for the TLS 1.3 standard candidate," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 483–502.

[25] V. Bernat, *TLS Computational DoS Mitigation*. Vincent Bernat, 2018. [Online]. Available: https://vincent.bernat.ch/en/blog/2011-ssl-dos-mitigation

[26] Transparency Market Research. (2017). *Software Defined Perimeter (SDP) Market*. [Online]. Available: https://www.transparencymarketresearch.com/report-toc/download/thanks/16916

[27] J. Koilpillai. (2016). *Software Defined Network (SDN) or Software Defined Perimeter (SDP). What's the Difference?* [Online]. Available: http://www.waverleylabs.com/software-defined-network-sdn-or-software-defined-perimeter-sdp-whats-the-difference/

[28] M. Rash. (2016). *Single Packet Authorization: A Comprehensive Guide to Strong Service Concealment With FWKNOP*. [Online]. Available: http://www.cipherdyne.org/fwknop/docs/fwknop-tutorial.html#install-fwknop

[29] K. Griffith. (2018). *Software-Defined Perimeter Remains Undefeated in Hackathon*. [Online]. Available: https://www.sdxcentral.com/articles/news/software-defined-perimeter-remains-undefeated-in-hackathon/2015/08/

[30] A. Iqbal, U. Javed, S. Saleh, J. Kim, J. S. Alowibdi, and M. U. Ilyas, "Analytical modeling of end-to-end delay in OpenFlow based networks," *IEEE Access*, vol. 5, pp. 6859–6871, 2017.

AHMED SALLAM received the B.Sc. degree in computer science from Suez Canal University, Egypt, the M.Sc. and Ph.D. degrees of Engineering in computer science and technology from Hunan University, China, in 2010 and 2013, respectively. He currently holds postdoctoral position at Western University, Canada.

AHMED REFAEY received the B.Sc. and M.Sc. degrees from Alexandria University, Egypt, in 2003 and 2005, respectively, and the Ph.D. degree from Laval University, Canada, in 2011. He is currently an Assistant Professor with Manhattan College and an Adjunct Research Professor with Western University. Previously, he held various positions including, a Senior Systems Architect, Mircom, from 2013 to 2016, and a Postdoctoral at ECE Department, Western University, from 2012 to 2013, and a Researcher at the LRTS Laboratory, from 2007 to 2011. His research interests include adaptive communication systems and networks security.

ABDALLAH SHAMI received the B.E. degree in electrical and computer engineering from Lebanese University, in 1997, and the Ph.D. degree in electrical engineering from the City University of New York, in September 2002. In 2002, he joined the Department of Electrical Engineering, Lakehead University, Thunder Bay, ON, Canada, as an Assistant Professor. Since July 2004, he has been with Western University, where he is currently a Professor and an Acting Chair with the ECE Department. His research interests include network optimization and cloud computing.