Software-Defined Perimeter (SDP): State of the Art Secure Solution for Modern Networks

Abdallah Moubayed, Ahmed Refaey, and Abdallah Shami

Abstract

The boom in the evolution and adoption of new technologies, architectures, and paradigms such as cloud computing, SDN, and NFV in recent years has led to a new set of security and privacy challenges and concerns. These challenges/concerns include proper authentication, access control, data privacy, and data integrity, among others. SDP has been proposed as a security model/framework to protect modern networks in a dynamic manner. This framework follows a need-to-know model where a device's identity is first verified and authenticated before gaining access to the application infrastructure. In this article, a brief discussion of the security and privacy challenges/concerns facing modern cloud-based networks is presented along with some of the related work from the literature. The SDP concept, architecture, possible implementations, and challenges are described. An SDP-based framework adopting a client-gateway architecture is proposed with its performance being evaluated using a virtualized network testbed for an internal enterprise scenario as a use case. To the best of our knowledge, no previous work has provided a quantitative performance evaluation of such a framework. Performance evaluation results show that the SDP-secured network is resilient to denial of service attacks and port scanning attacks despite needing longer initial connection setup time. The achieved results confirm the promising potential of SDP as a security model/framework that can dynamically protect current and future networks.

Introduction

In recent years there has been a boom in the evolution and proliferation of technology in our daily lives. The number of smart-phones, mobile-connected wireless devices, social networks, and sensors being used has grown substantially due to the emergence of the concept of Internet of Things (IoT). It was predicted by the National Cable & Telecommunications Association (NCTA) that the number of IoT devices will reach approximately 50 billion by the year 2020 [1]. To handle the increasing number of devices, several technologies and paradigms have been proposed. At the forefront of these technologies and paradigms are cloud computing, software-defined networking (SDN), and network function virtualization (NFV). Cloud computing has become a main component of the current technology landscape with more than 93 percent of organizations using cloud services in some shape or form [2]. As the cornerstone of Internet-of-Things (IoT)-based infrastructures, cloud computing has advanced a wide variety of applications such as patient monitoring and smart cities [3]. Projections performed by Microsoft predicts that the market size of cloud computing will reach 156.4 billion dollars by the year 2020 [4]. Similarly, the global market for SDN and NFV is expected to reach 54 billion dollars by the year 2022 [5, 6].

However, the adoption of such technologies and architectures has presented a new set of challenges, in particular those having to do with security. For example, McAfee reported that 52 percent of the respondents surveyed for their report indicated that they tracked a malware infection to a Software-as-a-Service (SaaS) application [2]. Moreover, it was reported that more than 6.1 million DDoS campaigns occurred in 2017 with both the Melbourne IT registrar attack and DreamHost attack being the most prominent [7]. Related challenges include having inadequate trust and authentication models, vulnerability to jamming and sniffing attacks, possibility of data loss or modification, and the possibility of information leakage [8, 9, 10]. Furthermore, other notable challenges facing such deployments are their vulnerability to man-in-the middle attacks, having inadequate access control measures, and the possibility of network intrusion [9, 11]. As a result, an exploration into new security measures to protect cloud-based networks has commenced, because traditional perimeter defense techniques have proven to be inadequate in protecting the infrastructure from network attacks [12, 13].

A promising solution is a software-defined perimeter (SDP), the concept proposed by the Cloud Security Alliance (CSA) as a security model/framework that has the potential to protect networks in a dynamic manner [12, 13]. This concept was developed based on the Global Information Grid (GIG) Black Core network initiative proposed by the Defense Information Systems Agency (DISA) [14]. This model follows a need-to-know model where the device's/application's identity is first verified and authenticated before it is granted access to the application infrastructure [12, 13]. Essentially, the infrastructure becomes "black," meaning, it is undetectable by infrastructures unauthorized to see it [12, 13]. This in turn can help mitigate many network-based attacks such as server scanning, denial of service, password cracking, man-in-the middle attacks, and many others [12, 13].

In this article, the security challenges/concerns

Digital Object Identifier:
10.1109/MNET.2019.1800324

The authors are with Western University, London, Ontario; Ahmed Refaey is also with Manhattan College. Abdallah Shami is the corresponding author.

0890-8044/19/$25.00 © 2019 IEEE

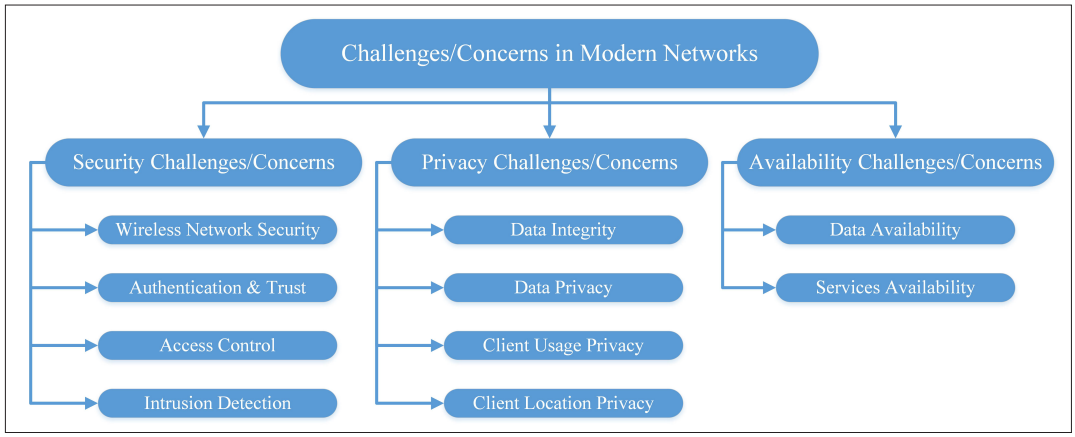

FIGURE 1. Challenges/concerns in modern networks.

facing modern networks are presented. Moreover, the SDP framework- as a potential solution- is described in more detail in terms of its concept, architecture, and possible implementations. Also, the anticipated challenges facing the SDP framework are also enumerated. Furthermore, an SDP-based framework adopting a client-gateway architecture is proposed. The performance of the framework is evaluated using an internal enterprise scenario in terms of connection setup time and network throughput under two types of network attacks, namely denial of service attack and port scanning attack. To the best of our knowledge, no previous work provides a quantitative performance evaluation of such a framework. Therefore, the contributions of this article can be summarized as follows:

- Propose the SDP framework as a potential security solution for modern networks in terms of its concept, architecture, and possible implementations.
- Evaluate and analyze the performance of the SDP framework in an internal enterprise scenario using a virtualized network testbed.

The remainder of this article is organized as follows. The following section discusses the security challenges/concerns and describes the SDP concept, architecture, and possible applications as a new proposed solution. We then present some of the challenges facing the SDP framework. Following that we describe the SDP-based framework, and then we evaluate the performance of the proposed SDP client-gateway framework to secure an internal enterprise network. The last section concludes the article and presents some future directions.

Software Defined Perimeter (SDP)

Despite the many advantages provided by adopting innovative architectures such as both SDN-based architectures and fog computing-based architectures to enhance the connection between edge and cloud computing, several challenges/concerns arise in such architectures [8–11]. As shown in Fig. 1, these challenges/concerns can be divided into three main categories: security challenges/concerns, privacy challenges/concerns, and availability challenges/concerns. More specifically, these categories include challenges such as authentication, access control, data integrity and privacy, and data availability. This results in added pressure on providers to offer frameworks and services that address these challenges. For interested readers, further details are available in [8–11].

Software-defined perimeter (SDP) is a potential solution to tackle many of the security and privacy challenges facing future networks. SDP was originally proposed by the Cloud Security Alliance (CSA) as a security model/framework having a dynamic ability to protect networks [12, 13]. This was part of the Global Information Grid (GIG) Black Core network initiative proposed by the DISA [14]. It adopts a need-to-know model where the device's/application's identity is granted access to the application infrastructure upon first being verified and authenticated [12, 13]. Due to this selective process, the infrastructure is referred to as "black." This is because it cannot be detected by users who are unauthorized to see it [12, 13]. As a result, this can effectively mitigate many network-based attacks including server scanning, denial of service, and man-in-the middle attacks, among many others [12, 13]. In what follows, the SDP concept, architecture, and possible implementations are discussed in more details.

Concept

The SDP concept is built on the notion of providing application/service owner(s) with the power to deploy perimeter functionality as needed to protect their servers. This is done by adopting logical components in place of any physical appliances. These components are controlled by the application/service owner(s) and serve as a protection mechanism. The SDP architecture only provides access to a client's device after it verifies and authenticates its identity. Such an architecture has been adopted by multiple organizations within the Department of Defense in which servers of classified networks are hidden behind an access gateway. The client must first authenticate to this gateway before gaining visibility and access to the server and its applications/services. The aim is to incorporate the logical model adopted in classified networks into the standard workflow (presented in more detail below). Hence, the SDP architecture leverages the benefits of the need-to-know model while simultaneously eliminating the need for a physical access gateway. The general concept is that the client's devices/applications are first authenticated and authorized before cre-

The proposed framework adopts a dynamic firewall policy at the gateway by vigorously adding and removing rules to allow authenticated and authorized users to access the protected applications and services. This does not add to or change the existing system's complexity.

ating encrypted connections in real-time to the requested servers.

The SDP architecture is composed of and relies on five separate security layers.

Single Packet Authentication (SPA): SPA is the cornerstone of device authentication. The SDP uses this SPA to reject traffic to it from unauthorized devices. The first packet is cryptographically sent from the client's device to the SDP controller where the device's authorization is verified before giving it access. The SPA is then again sent by the device to the gateway to help it determine the authorized device's traffic and reject all other traffic.

Mutual Transport Layer Security (mTLS): Transport layer security (TLS) was originally designed to enable device authentication and confidential communication over the Internet. Despite the fact that the standard offers mutual device authentication, it has typically only been used to authenticate servers to clients. However, the SDP utilizes the full power of the TLS standard to enable mutual two-way cryptographic authentication.

Device Validation (DV): Device validation adds an extra layer of security by ensuring that the cryptographic key used is held by the proper device, because mTLS only proves that the key has not expired nor has it been revoked. However, it cannot prove that it has not been stolen. Therefore, DV verifies that the device belongs to an authorized user and is running trusted software.

Dynamic Firewalls: In contrast to traditional static firewalls that can have hundreds or thousands of rules, dynamic firewalls have one constant rule which is to deny all connections. The SDP adopts a dynamic firewall policy at the gateway by vigorously adding and removing rules to allow authenticated and authorized users to access the protected applications and services.

Application Binding (AppB): Application binding refers to the process of forcing authorized applications to use the encrypted TLS tunnels created by the SDP. This is done after the device and the user are properly authenticated and authorized. This ensures that only authorized applications can communicate through the tunnels while unauthorized applications are blocked.

These protocols combined make it extremely challenging for malicious users and attackers to access protected applications and services. Consequently, the SDP framework can address many of the aforementioned security, privacy, and availability challenges, including authentication and trust, access control, data privacy, data availability, and services availability.

It is worth mentioning that the complexity of this framework is mainly based on that of the hash function used as part of the encryption/decryption of the SPA packet since the other four security layers of the framework already exist. For example, dynamic firewalls are used in layer 3. In contrast to traditional static firewalls deployed in existing systems which can have hundreds or thousands of rules, dynamic firewalls have one constant rule which is to deny all connections. The proposed framework adopts a dynamic firewall policy at the gateway by vigorously adding and removing rules to allow authenticated and authorized users to access the protected applications and services. This does not add to or change the existing system's complexity.

Architecture

The SDP framework's architecture consists of three main components [12, 13].

SDP Controller: The SDP controller is the central element in the SDP framework. It is responsible for all the control messages exchanged by functioning as a trust broker between the initiating SDP host and backend security controls. This includes issuing certificates and authenticating devices (both initiating and accepting hosts). Moreover, it determines the services that each initiating host is authorized to access in the accepting host. Furthermore, it helps configure both the SDP initiating host and accepting host in real time to create the mutual TLS tunnel [12, 13].

SDP Client/Initiating Host (IH): SDP Initiating Hosts are the SDP-enabled clients that submit a request to connect to a service or application. This request is submitted to the SDP controller which will authenticate the IH by requesting information about hardware or software within the IH. Once authentication is completed using the previously issued certificate, a mutual TLS tunnel is created that connects the IH to the server or application for which it has authorization. This helps improve the access control since the IH only gets access to the server or application after being properly authenticated [12, 13].

SDP Accepting Hosts (AH): On the other hand, the SDP Accepting Host (AH) is the device instructed to accept authorized services or applications. It is originally set up to reject all incoming packets and requests from all hosts except the SDP controller. The SDP host is typically protected by an SDP gateway which acts as the protector. This gateway is the termination point for the mutual TLS tunnel from the IH. This is done after the SDP controller provides the gateway with the verified and authenticated IH's IP address and certificates [12, 13].

Potential Applications

The SDP architecture can be implemented in several ways despite the workflow remaining the same. This includes client-to-gateway, client-to-server, server-to-server, and client-to-server-to-client, making it suitable for a variety of applications as shown in what follows.

Core Networks: Despite the appeal of NFV for core networks in terms of dynamically creating, managing, and adjusting security zones due to the automated placement of virtualized firewalls and creation of dedicated software firewalls on-demand, it is prone to several security vulnerabilities such as resource exhaustion, service hijacking through the self-service portal, and service insertion. For example, due to the sharing of the physical servers' resources among different VMs, severe resource exhaustion can occur that

can negatively impact VM availability. An SDP framework can tackle this by defining and implementing a VM throttling detection mechanism by the controller and having it instantly propose a remedy. Another example is the service hijacking through the self-service portal.

Once again, the SDP proves its vital role in eliminating this risk by using administrative controls selectively, based on users' roles and needs.

Mobile Networks: The emergence of Internet of Things (IoT) and machine-to-machine (M2M) technologies due to the deployment of a billion-plus smart connected devices worldwide have positively impacted businesses. However, the protocols used in IoT raises a real concern regarding aspects in security. Despite most of these protocols being designed for wireless sensor networks, they have had a sudden and unanticipated role in the core of the IoT solution. Message Queuing Telemetry Transport (MQTT) is a good example of a viable transport layer for wireless networks in the IoT era. When the MQTT protocol was designed, its security was not a priority because it had been typically deployed in secure, backend networks for application-specific purposes. Moreover, due to the two-way handshake mechanism adopted, the protocol is vulnerable to hijacking and man-in-the-middle attacks. The SPA packet mechanism used in an SDP framework can replace the username/password login mechanism. Moreover, the "blackening" of receiving devices is an added security layer to protect against hackers since they become undetectable to them. Also, all SPA packets are encrypted and authenticated with an HMAC signature. This leads to malicious users having to first steal the SDP credentials to be able to spoof an SPA packet. The combination of this with the logging of all validated SPA packets by the SDP gateway creates a broker immune to faked and replayed SPA packets.

Internal Enterprises Networks: Given the continued adoption of the bring your own device (BYOD) concept and the increased number of mobile workers, the traditional perimeter concept becomes obsolete and uncontrollable. Hence, securing the internal enterprise network became a tedious task with system administrators having to take aggressive decisions just to avoid the threats. Virtual private networks (VPNs) have provided secure access to virtual local area networks for remote users. However, such legacy VPNs were designed for the 1990s networks and thus are obsolete due to their lack of agility in protecting digital businesses [15]. In contrast, an SDP framework allows organizations and enterprises to keep cloud resources dark to unauthorized users. This helps protect against a variety of attacks including network flood attacks, brute-force attacks, and TLS vulnerabilities. Hence, by having a "dark net" around the servers, an SDP framework can facilitate the management and security of organizations' cloud resources. Therefore, SDP can provide an ideal VPN solution for any size organization as it provides cloud-based controllers, gateways, and up-to-date software defined security perimeters to deliver secure, dynamic virtual connectivity.

This work focuses on the internal enterprise case and proposes an SDP-based framework to protect it. Since the SDP framework adopts a zero-trust architecture by authenticating and verifying a host for every session, it can address the potential lateral threat model often found in such environments. The performance of the framework is evaluated for both the SDP and non-SDP case to show the potential of SDP as a security model and alternative to VPNs for internal enterprise networks.

Since the SDP framework adopts a zero-trust architecture by authenticating and verifying a host for every session, it can address the potential lateral threat model often found in such environments. The performance of the framework is evaluated for both the SDP and non-SDP case to show the potential of SDP as a security model and alternative to VPNs for internal enterprise networks.

SDP Open Challenges

Despite the many advantages that the SDP architecture provides with respect to protecting against various network security breaches and attacks, it faces several challenges.

Possible Network Disruption: Since the SDP architecture is different than traditional security measures deployed in networks, integrating a complete SDP solution can lead to network and infrastructure disruption. This can be problematic due to the size of the services that may be off during such a disruption.

Configuration Updates: A second challenge is updating all the applications and system configurations in such a manner that they become aware of the SDP. This will allow them to access the workflow and the secure tunnels created within the SDP.

Controller Vulnerability: Because the SDP controller has a major role in the architecture and hence the overall security of the network, protecting it becomes paramount. Moreover, given that it is a possible centralized point of failure, it needs to be made highly available and secured.

Proposed Framework

As mentioned earlier, the SDP is composed of three main components: client, gateway and controller. For the purpose of this work, the SDP architecture proposed and implemented is the client-gateway architecture as shown in Fig. 2 which can faithfully describe an internal enterprise network scenario. In this case, the gateway also acts as the application server and hosts the service to be accessed. The functions of the three components are as described below.

Controller: The controller is the main component of SDP. It contains the details of the authorized clients and servers, provides the details of rules to the gateway and controls the authentication of each component. The controller proposed in this work makes use of a database for all the above purposes. The database contains the details of all the hosts involved, which is then sent to the gateway. It authenticates these hosts with the help of certificates.

Gateway: The gateway enforces the rules which prevent any unauthorized access to the service hidden behind it. By default, the gateway blocks all traffic. However, once the controller provides the list of authorized initiating (clients) and accepting hosts (servers) and the list of services, it sets up rules which allow a connection to

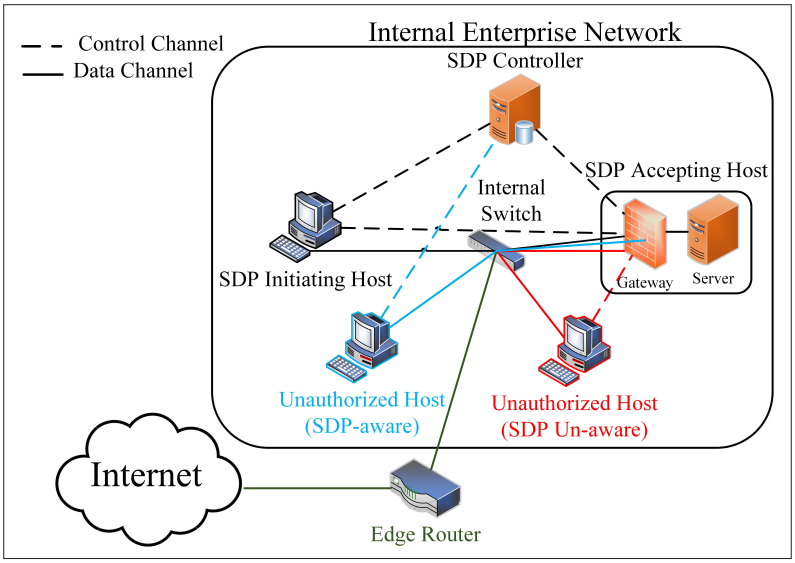

FIGURE 2. Proposed SDP framework for internal enterprise network.

be established between the two while preventing all other traffic. This includes any attempt by the authorized hosts to establish a connection to a service they are not authorized to access. In this work, these rules are set up using the iptables.

Client: The client is the machine trying to access a service. In SDP, the client first connects to the controller and informs it about the service it wishes to access. Once the verification is complete, it attempts to connect to the service hidden behind the gateway. The gateway will allow the connection request to go through and thus, the client-server data transfer can take place. Ideally, once the connection is established it should not be reset until specifically requested.

The whole process of SDP, as shown in Fig. 3, involves the following steps

- The gateway initiates a TLS connection to the controller and sends an SPA packet.
- The controller verifies the gateway using a certificate present in the gateway.
- The controller establishes a mTLS secured connection between itself and the gateway.
- The controller then proceeds to send all the information about the initiating and accepting hosts as well as the authorized services to the gateway.
- The client initiates a TLS connection to the controller and sends its own SPA packet.
- The controller verifies the client using a certificate present in the client.
- The controller establishes a mTLS secured connection between itself and the client.
- The client sends another encrypted authentication SPA packet to the gateway.
- The packet is decrypted with the keys provided by the controller.
- The information in the decrypted packet is then crosschecked with the information it received from the controller.
- The gateway sets up the corresponding firewall rules after a positive crosscheck.
- The client attempts to connect to the service.
- The connection is established once the gateway allows the connection request to pass and data transfer can take place.

Performance Evaluation

Virtualized Network Testbed Description:

Waverly Labs' OpenSDP project was used to implement the proposed framework with the previously discussed components being programmed using C, Python, and NodeJS. Each component was set up on a separate virtual machine with a fourth acting as an unauthorized host.

A mySQL database was implemented in the controller to form the basis of the list of authorized hosts and services. Within this database, several tables have been created including: **sdpid**, **service**, **service_gateway**, and **sdpid_service**. The first table contains the details of each component within the SDP-enabled setup, the SDP ID assigned to each component, their decryption and HMAC keys, and timestamps of the last update done on the keys and so on. The second table is used to define the service ID for each service authorized to be accessible by a host SDP ID, while the third defines the gateway ID which protects a particular service. Finally, the fourth table specifies the mapping of the service ID to the protocol and port number of the service. The controller creates a hash of this information and sends it to the gateway to be used as a verification mechanism. Once the authentication is complete, a SSL connection is established, and the keys are updated. These connections are persistent as long as both sides keep them open. The controller also informs all gateways about any changes made in the SDP network and immediately transmits it to them.

On the other hand, the gateway sets up the forwarding rules for successful transmission of authorized packets by making use of the iptables. Another available option is *firewalled* wherein the gateway makes use of the FireWall KNock OPerator (FWKNOP) mechanism for packet authorization. Using this mechanism, the client sends the Single Packet Authorization (SPA) packet, a specially crafted encrypted packet, to the gateway. To authenticate the packets, the gateway first connects to the controller and downloads all the necessary information. Note that the firewall's default rule is drop-only. Iptables consists of three tables: INPUT, OUTPUT and FORWARD. Their functions are self-explanatory. FWKNOP creates another table called FWKNOP_INPUT which contains the details of the source and destination IP of the authorized packet. It also contains details of the protocol and port number of the service. The gateway attempts to decode the packet as soon as it receives it with the keys obtained from the controller. Once the packet is decoded, it verifies the details such as the SDP ID of the sending host and the service ID requiring access, among others. Ensuing the verification of the packet, the gateway then creates a rule in iptables that allows the client to connect to the service. This rule is present for a limited amount of time (default value of 10s). The established connection is a mutual TLS connection and persists even after the expiry of the rule. The gateway goes back to its original state of blocking all packets. As soon as a packet from an authorized machine arrives, it opens the firewall, allows the packet to pass through it and then closes the firewall again.

Criterion		Without SDP	With SDP
Connection setup time		0.09 sec	1.02 sec
Average network throughput	TCP	6.86 Mb/s	1.51 Gb/s
	UDP	1.7 kb/s	8.8 Mb/s

TABLE 1. Performance evaluation results.

In contrast, the client creating the SPA packet to send it to the gateway for verification includes the following fields within the packet:

- SDP ID
- Service ID
- Gateway IP
- Timestamp
- Random 16-byte data

The first two fields are used to disclose the client's identity and inform the gateway of the service it wishes to connect to. The third field is the specific gateway's IP address that the client wishes to establish a connection to. The timestamp, used to verify the "age" of a packet, is another authentication mechanism of SDP. If the timestamp of the packet exceeds a pre-determined limit, then the packet is rejected. This, combined with the random 16-byte data, is used to ensure that replay attacks cannot be used on the network. The gateway establishes rules in itself as soon as it verifies the SPA packet to allow packets from the client to travel to the destination service. Then the client establishes a connection to the service and data transfer can take place.

Results Analysis and Discussion

The connection setup time and the overall network throughput are the two metrics considered when evaluating the performance of the proposed framework. This is done for both the SDP and non-SDP scenarios. Moreover, two attacks are simulated, namely a distributed denial of service (DDOS) attack and a port scanning attack. The DDoS attack was chosen as it represents a threat to the data and services availability for modern networks. Similarly, the port scanning attack represents a threat to the data privacy. Therefore, these two attacks were considered as they represent two threats and concerns within modern networks that the SDP framework aims to address. The results reported are the average of ten different runs.

Connection Setup Time: The first performance metric considered is the connection setup time. This metric refers to the time taken by a client to connect to a service with and without SDP. In this experiment, the time needed to setup a SSH connection between the client and the gateway is determined. The results are shown in Table 1. It can be observed that a higher setup time is needed when adopting SDP. This is expected given the decryption and verification process of the SPA packet as well as the setup rules for the firewalls needed before the client connection can attempt to go through. On the other hand, none of these steps occur in the non-SDP case, therefore allowing for a significantly faster connection process.

Distributed Denial of Service Attack: The second performance metric considered is the network throughput which refers to the measured network speed at the port where a connection is being established in case of a DDoS attack, with and without SDP. To simulate the DDoS attack, one VM is set up as an authorized client, while the other as an unauthorized client. The unauthorized client runs a software to emulate the DDoS attack by attempting to establish multiple connections to the port and hence attempting to consume all of the network bandwidth available. The experiment involved sending both TCP and UDP traffic. For the TCP case, the available bandwidth was kept at 2 Gb/s whereas for UDP, the bandwidth was kept at 10Mb/s. This is because when using UPD in such a setup, it is extremely difficult to consume the entire 2 Gb/s bandwidth. The average network throughput results are shown in Table 1.

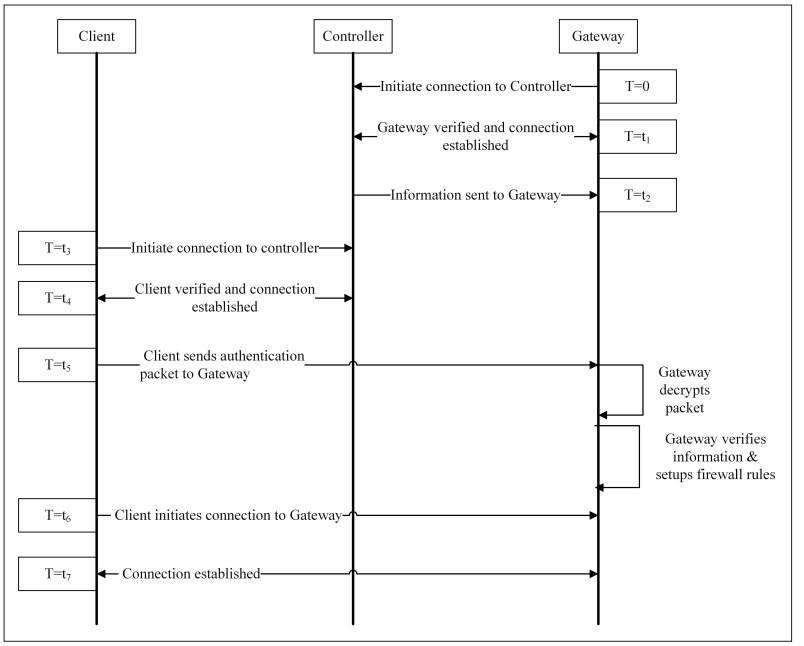

FIGURE 3. Proposed framework's process workflow.

The connection was attempted on port 5000. Ideally, the SDP is assumed to block all unauthorized traffic and only allow authorized traffic. This keeps the bandwidth value as the one set up for the connection. However, this is not the case in practice. The firewall opens up once it receives an authorized packet to allow it to pass while also allowing all other traffic through before it closes up again. Hence, the available bandwidth to the client slightly decreases. However, the results clearly show the impact of SDP when using both TCP and UDP protocols. When using TCP, the average throughput without SDP falls to around 0.343 percent of the available bandwidth, whereas in the case of SDP, it hovers around 75.5 percent. Similarly, when using UDP, the average throughput without SDP falls to around 0.017 percent while that of the SDP is around the 88 percent range. This further highlights the fact that SDP works fairly well in protecting networks from DDoS attacks.

Figure 4 shows the impact of a DDoS attack on the performance of the network without SDP by plotting the throughput over time. It is observed that the data transfer could not happen in one shot due to the DDoS attack clogging up the network. It is worth noting that since the

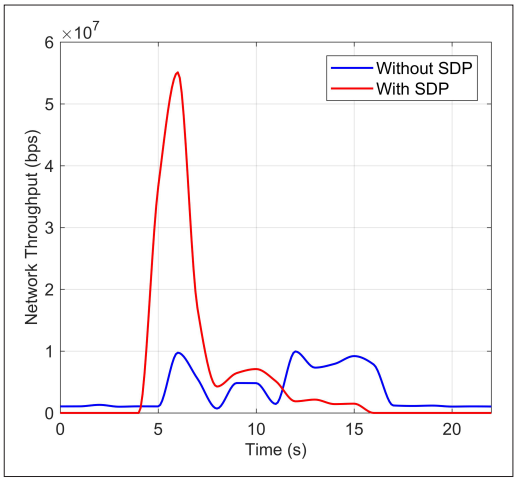

FIGURE 4. Network throughput over time.

```
Nmap scan report for 192.168.74.4
Host is up (0.00015s latency).
Not shown: 999 closed ports
PORT STATE SERVICE
22/tcp open ssh
```

a: Without SDP

```
Nmap scan report for 192.168.74.4
Host is up (0.00030s latency).
All 1000 scanned ports on 192.168.74.4 are filtered
```

b: With SDP

FIGURE 5. Port scanning attack results: a) without SDP; b) with SDP.

attack is being perpetrated from one host, its overall impact may be quite limited. However, it is enough to disrupt the data transfer by not allowing the full transfer to take place. The average throughput during the entire data transfer process is around 6.86 Mb/s. This is a significant drop from the 2 Gb/s bandwidth actually available within the network. In contrast, the positive impact of adopting SDP is highlighted. In this case, the data transfer was completely successful and the average bandwidth during the process is 1.51 Gb/s. Since the DDoS attack is running from a single host, the impact on the bandwidth is minimal. Note that if the same attack was launched with multiple hosts, the bandwidth may fall even further. However, it will still be enough for the application to function normally.

Port Scanning Attack: The second attack simulated is the port scanning attack. This attack involves running a check and listing all the available ports. Such an attack is an active form of attack, where using a tool can be used to discover the hosts and services on a network. It can further be used to execute attacks on specific ports such as port 80 (HTTP) and so on. This experiment uses nmap, a free port scanning utility. Nmap ran from the unauthorized host on the gateway, with and without SDP. The results of this attack are presented in Figs. 5a and 5b for the non-SDP and SDP scenarios, respectively. Note that the scan ran on ports 1-1000, covering some of the more important services such as HTTP, FTP, SSH and so on.

It can be clearly seen that without SDP, the scan report shows that an SSH connection is open and available. In contrast, SDP gives no information whatsoever about the number of open ports and the service hosted on them. This enforces the security of the application and confirms the unavailability of access to unauthorized clients.

Conclusion

This article described the SDP framework as a potential solution to the different security challenges/concerns facing modern networks in terms of its concept, architecture, and possible applications. Also, the challenges facing the SDP framework were briefly presented and discussed. An SDP-based framework adopting a client-gateway architecture was proposed and its performance was evaluated using an internal enterprise scenario in terms of connection setup time and network throughput under two types of network attacks, namely distributed denial of service attack and port scanning attack. The performance evaluation of the implemented virtualized network testbed showed that an SDP-secured network is more resilient to port scanning and distributed denial of service attacks by not providing any information and maintaining a high average network throughput respectively despite needing a longer time to initially establish the connection. These results confirm the promising potential of SDP in protecting current and future networks.

Despite the early promise of SDP, more open research areas exist. In particular, exploring how to integrate SDP with other paradigms such as SDN and NFV is essential since these paradigms will play a significant role in future networks.

Acknowledgments

The authors would like to thank Mr. Palash Kumar for his help in the implementation of the proposed framework. His contributions have been extremely valuable for the completion of this work.

References

[1] National Cable & Telecommunications Association (NCTA), "Behind The Numbers: Growth in the Internet of Things," Mar. 2015; available: http://www.ncta.com/whats-new/behind-the-numbers-growth-in-the-internet-of-things
[2] McAfee, "Building Trust in a Cloudy Sky: The State of Cloud Adoption and Security," Jan. 2017; available: http://www.mcafee.com/us/solutions/lp/cloud-security-report.html
[3] C. Esposito *et al.*, "Challenges of Connecting Edge and Cloud Computing: A Security and Forensic Perspective," *IEEE Cloud Computing*, vol. 4, no. 2, Mar. 2017, pp. 13–17.
[4] Microsoft, "Digital Transformation at Work: Empowering Together," Mar. 2017.
[5] N. Bannerman, "SDN and NFV Market to be Worth $54 Billion by 2022," Aug. 2017.
[6] H. Hawilo *et al.*, "NFV/SDN-Based vEPC Solution in Hybrid Clouds," *Proc. 2018 IEEE Middle East and North Africa Commun. Conf. (MENACOMM)*, April 2018, pp. 1–6.
[7] A. Moubayed *et al.*, "DNS Typo-Squatting Domain Detection: A Data Analytics & Machine Learning Based Approach," *Proc. IEEE Global Commun. Conf. (GLOBECOM'18)*, Dec. 2018, pp. 1–7.
[8] G. Kulkarni *et al.*, "Security Aspects in Cloud Computing," *Proc. IEEE Int'l. Conf. Computer Science and Automation Engineering (CSAE'12)*, June 2012, pp. 547–50.
[9] S. Yi, Z. Qin, and Q. Li, "Security and Privacy Issues of Fog Computing: A Survey," *Proc. Int'l. Conf. Wireless Algorithms, Systems, and Applications (WASA'15)*, Springer International Publishing, 2015, pp. 685–95.
[10] R. von Solms and J. van Niekerk, "From Information Security to Cyber Security," *Computers & Security*, vol. 38, 2013, Cybercrime in the Digital Economy, pp. 97–102; available: http://www.sciencedirect.com/science/article/pii/S0167404813000801

[11] I. Stojmenovic and S. Wen, "The Fog Computing Paradigm: Scenarios and Security Issues," *Proc. Federated Conf. Computer Science and Information Systems (FedCSIS'14)*, Sept. 2014, pp. 1–8.
[12] Software Defined Perimeter Working Group-Cloud Security Alliance (CSA), "Software Defined Perimeter," Dec. 2013; svailable: http://downloads.cloudsecurityalliance.org/initiatives/sdp/Software Defined Perimeter.pdf
[13] P. Kumar *et al.*, "Performance Analysis of SDP for Secure Internal Enterprises," *Proc. IEEE Wireless Commun. Networking Conf. (WCNC'19)*, Apr. 2019.
[14] Department of Defense, "Department of Defense Global Information Grid Architectural Vision," June 2007.
[15] SAIFE, "SAIFE Extends the Software Defined Perimeter," June 2017; available: https://www.saife.io/press-releases/saife-extends-software-defined-perimeter/

Biographies

Abdallah Moubayed received his B.E. degree in electrical engineering from the Lebanese American University, Beirut, Lebanon in 2012, his M.Sc. degree in electrical engineering from King Abdullah University of Science and Technology, Thuwal, Saudi Arabia in 2014, and his Ph.D. in electrical & computer engineering from the University of Western Ontario in August 2018. Currently, he is a postdoctoral associate in the Optimized Computing and Communications (OC2) lab at the University of Western Ontario. His research interests include wireless communication, resource allocation, wireless network virtualization, performance and optimization modeling, machine learning and data analytics, computer network security, cloud computing, and e-learning.

Ahmed Refaey received his B.Sc. and M.Sc. degrees from Alexandria University, Egypt in 2003 and 2005, respectively, and a Ph.D. degree from Laval University, Quebec, Canada in 2011. Currently, he is an assistant professor at Manhattan College as well as an adjunct research professor at The University of Western Ontario. Previously, his positions included: Sr. Embedded Systems Architect, R&D group, Mircom Technologies Ltd from 2013-2016; and as a Postdoctoral Fellow in the ECE Department, The University of Western Ontario from 2012-2013. His research interests include adaptive communication systems and networks security, IoT/emerging communications/computing technologies and applications, and embedded systems/FPGA prototypes implementation.

Abdallah Shami is a professor with the ECE Department at Western University, Ontario, Canada. He is the Director of the Optimized Computing and Communications Laboratory at Western University (https://www.eng.uwo.ca/oc2/). He is currently an associate editor for *IEEE Transactions on Mobile Computing*, *IEEE Network*, and *IEEE Communications Surveys and Tutorials*. He has chaired key symposia for IEEE GLOBECOM, IEEE ICC, IEEE ICNC, and ICCIT. He was the elected Chair of the IEEE Communications Society Technical Committee on Communications Software (2016-2017) and the IEEE London Ontario Section Chair (2016-2018).