On SDPN: Integrating the Software-Defined Perimeter (SDP) and the Software-Defined Network (SDN) Paradigms

Michael Lefebvre, Daniel W. Engels, Suku Nair
AT&T Center for Virtualization
Southern Methodist University
Dallas, TX, USA
{lefeb, dwe, nair}@smu.edu

2022 IEEE Conference on Communications and Network Security (CNS) | 978-1-6654-6255-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/CNS56114.2022.9947267

***Abstract*—In this paper, we introduce the Software-defined Perimeter Network (SDPN) that is a single virtual Zero Trust overlay network that can provide perimeter-like functionality across the Internet. Modern networks increasingly need to integrate virtual network components that exist outside of the organization and often exist within multiple different cloud-based systems. These virtual network components are operated most efficiently when they operate within a single network management domain, traditionally achieved via castle-and-moat network perimeter.**

SDPN integrates the multi-plane management model (data, control, and application planes) and core functionality of the Software-defined Perimeter (SDP) and the Software-defined Network (SDN) paradigms with Zero Trust security principles to create a single virtual network perimeter. In the control plane, the SDPN controller combines SDP controller and SDN controller functionality into one logical controller. The SDPN controller also introduces trust as the trust anchor via a root certificate issued by a trusted third-party. At the data layer, servers, devices, and network functions are simplified into nodes. The SDPN controller manages every node in the perimeter via a control channel.

The SDPN framework provides for Zero Trust security both within and beyond a managed network domain while reducing the attack surface, securing network flows, and enabling simple network management.

***Index Terms*—Software-defined Perimeter, Software-defined Network, Zero Trust, Zero Trust Architecture, SDP, SDN, ZTA**

I. Introduction

Networks are moving to a virtual, dynamic, and distributed architecture that spans public networks and hybrid public-private clouds. The diversity and distribution of the logical network components has increased the importance of managing and securing these networks at scale, while maintaining services and functionality both within and between networks. Securing networks and services can no longer be considered physically constrained to one location.

Existing approaches to managing and securing virtual networks relies upon classic physical security network paradigms that separate enforcing trust and security *within* the network from enforcing trust and security *between* networks. Traditional networks have well-defined perimeters and separate services that make managing defense-in-depth security and the network tractable [1], [2]. When applied to virtual networks, however, this results in security gaps, uneven service protections, and complex operational management. The issues stem from the ethereal nature of virtual networks where the perimeter is not easily contained to a few exit nodes, when all nodes in a virtual network may, in fact, be perimeter nodes. This complicates management and security for virtual networks.

Traditional castle-and-moat network perimeter paradigms cannot address these future network architectures. With the traction of Zero Trust as a future security model [3], the traditional perimeter must evolve to a *virtual perimeter* model that is physically distributed, logically contained, and locally empowered. That is, each node logically contained inside a perimeter must be able to inherit and uniformly enforce perimeter policies.

To this end, we define the Software-defined Perimeter Network (SDPN), an integration of Software-defined Perimeter (SDP) and Software-defined Networking (SDN) into a single virtualized network framework. SDN as a well-defined networking model excels at enforcing trust and security *within* a network. SDP as an emerging model defines an approach for enforcing trust and security *between* networks. Integrating the two software-defined models provides the ability to enforce trust and security *within and between* networks while fundamentally incorporating Zero Trust tenets.

The SDPN framework aligns with SDP, SDN, and Zero Trust models by maintaining separation of network and perimeter functionality into data, control, and application planes. At the control plane, the SDPN integrates the functionality of an SDP controller and an SDN controller into a logical SDPN controller that also introduces security functionality as a cryptographic trust anchor [4]. At the data plane, SDPN nodes are an abstraction of network functions, user devices, or servers and are identified as a perimeter node, host node, or service node, respectively. Data traverses between nodes via the data channel while the SDPN controller sends control messages via a control channel that is maintained with every node.

The SDPN framework defined herein is a viable network security framework for future network architectures. SDPN applications integrate with existing network protocols and are

978-1-6654-6255-6/22/$31.00 ©2022 IEEE

TABLE I
RELATIONSHIP AND COMPONENTS ACROSS ABSTRACTION LAYERS OF SDN, ZERO TRUST, SDP, AND SDPN.

Abstraction Layer	SDN	Zero Trust	SDP	SDPN
Application	SDN Applications	Zero Trust Applications	SDP Applications	SDPN Applications
Control	Controllers	Policy Decision Point (PDP)	Controllers	Controllers
Data	Switches	Policy Enforcement Point (PEP)	Hosts	Nodes

supplemented by SDPN control channel protocols that enable trusted network scaling and inter-network trust exchanges. This provides the ability to implement and manage virtual perimeter security at scale across distributed networks.

The remainder of this paper is organized as follows. We provide a high-level overview of SDN, SDP, and Zero Trust in Section II. We then present the SDPN framework in Section III. A practical case study of an SDPN and its new network sequence flows are illustrated in Section IV. A discussion of the SDPN framework and implementation follows in Section V. We draw the relevant conclusions in Section VI.

II. SDP, SDN, AND ZERO TRUST OVERVIEW

SDP and SDN emerged out of a need to address scale and increasing complexity. Both software-defined paradigms addressed these by abstracting functionality into two logical planes: data and control. Related applications built for the control plane led to the natural evolution of adding an application plane atop the control plane. Parallel to these software-defined models emerged the concept of Zero Trust as an overall paradigm for computer security, where trust enforcement and trust decisions also align with data and control planes, respectively [5]. The crosswalk of the relationships across these layers, with the associated components that comprise each layer, is outlined in Table I. A high-level overview of these three models follows, in chronological order of their origination.

A. Software-Defined Networking

Software-defined Networking is rooted in research that traces back to the 1990s [6], but gained traction circa 2007 onward based on an implementation at Stanford called “Ethane” [7], [8]. The fundamental principle is the separation of network management and functionality into two planes: control and data. This is achieved by extracting network logic and management functions to a centralized SDN controller in the control plane that manages one-to-many network switches. Network switches reside in the data plane and strictly operate as packet pushers that consult the controller if they do not know how to handle a packet. This paradigm shift provides the ability for centralized management of networks with an OS-like language. While SDN (and its real-world implications) continues to evolve [9], its impact has broad reach into almost every type of network deployment, including 5G/6G [10], automation [11], network elasticity [12], internet of things [13], and even vehicular communication [14]. Unsurprisingly, there is a large corpus of ongoing SDN research as cataloged in various comprehensive survey papers [15]–[17].

B. Zero Trust

Zero Trust was coined in 2010 by Forrester in response to traditional castle-and-moat network perimeter models where systems inside the perimeter were assumed to be trusted simply by nature of their placement inside the perimeter [18]. The Zero Trust paradigm shift is about zero *inherent* trust, meaning security policies should never be predicated on unchallenged inferences. Zero trust principles have since been introduced to drive the industry toward adoption [3], [19], even inspiring a White House decree for United States Federal Agencies to adopt Zero Trust architecture and principles [20].

In 2020, the United States National Institute of Standards and Technology (NIST) published ”Zero Trust Architecture” where it defined the concepts of a Policy Decision Point (PDP) and Policy Enforcement Point (PEP) [5]. The PDP is the control plane logic that determines policies (e.g., allow/deny) for PEPs to enforce in the data plane. Multiple data points such as user identity, geographic location, and device type provide data points for a PDP trust evaluation [19].

While Zero Trust has become an overloaded term since its inception, here we share a great working definition to baseline its vision:

> Zero Trust is a holistic model for securing network, application, and data resources, with a focus on providing an identity-centric policy model for controlling access. [3, p. 13]

C. Software-Defined Perimeter

Software-defined Perimeter (SDP) was first defined by the Cloud Security Alliance (CSA) in 2014 [21] as an abstraction of the network perimeter from a physical implementation to a logically defined perimeter. This paradigm was in response to the proliferation of network devices, coupled with increased mobility of these devices, leading to the dissolution of a traditional network perimeter. SDP was updated in 2022 to Specification 2.0 [22].

SDP consists of two logical components: hosts and controllers [22]. A controller defines the control plane that manages and enforces policy across the SDP. Hosts comprise the data plane and can be any computing device that can initiate or accept a network connection. Hosts are cataloged into two types: an Initiating SDP Host (IH)—commonly known as a client; and, the Accepting SDP Host (AH)—traditionally known as a server.

A cornerstone of SDP is the Single Packet Authorization (SPA) packet. The SPA is an individual cryptographic packet

that contains authentication[1] information within its payload [23]. Accepting Hosts (AHs) within an SDP are configured to drop all unsolicited inbound packets without first receiving a valid SPA. SPA is regularly used in an SDP to hide services on the network.

III. SDPN Framework

The SDPN is comprised of two logical components: controllers and nodes. SDPN aligns with existing SDN, SDP, and Zero Trust models by defining a data plane, control plane, and application plane. SDPN controllers provide the control layer governance over nodes that reside in the data layer, as outlined in Table I and depicted in Fig. 1. The SDPN controller combines the functionality provided individually by an SDP controller and an SDN controller into one logical controller. Applications in the application plane are built upon the control layer to provide logic and functionality.

At the SDPN data layer, SDP hosts and SDN switches are collapsed simply into nodes. Nodes may be of type host, service, or perimeter. Subtybes may be defined within these node categories to further delineate primary categories of functionality. This framework allows for future flexibility to accommodate new technologies that may regularly integrate with an SDPN (e.g., IoT sensor as a service node subtype). SDPN components, applications, and initial protocols are defined in the following subsections.

A. SDPN Components

There are four fundamental components of an SDPN:

- Controller: the SDPN authority that comprises the control plane and integrates the logic and functionality provided in traditional SDP and SDN controllers. A minimum of one SDPN controller is required to establish the trust anchor for the SDPN. The SDPN controller maintains a control channel with every node in the SDPN, providing a centralized management point of all nodes. It is the authority in establishing and managing the SDPN, including trust management, managing network flows, and defining Zero Trust policies. The SDPN controller can scale horizontally to multiple controllers via existing SDN controller approaches to fault-tolerance and resiliency [24], [25].
- Perimeter Node: any node that provides network functionality. This is any physical or virtual network function that enables the transmission or blocking of network packets. Sample subtypes include switch perimeter node (e.g., a traditional SDN switch), firewall perimeter node, or router perimeter node.
- Service Node: any node that provides a service (e.g., a server). Service nodes can be categorized into further subtypes such as web service node, directory service node, or file service node.
- Host Node: any node that primarily consumes services (e.g., a laptop or other end user device).

[1]Note: authentication and authorization are seemingly used interchangeably, but are two different access control concepts.

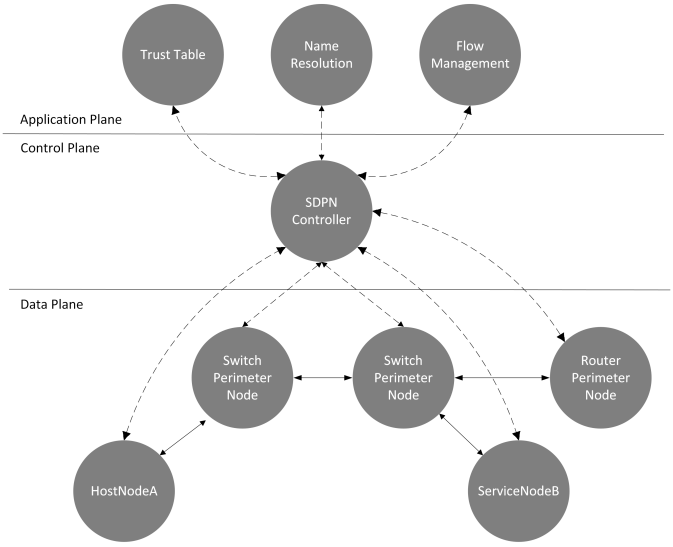

Fig. 1. A simple logical SDPN architecture. Control channels are represented by dashed lines; data channels by standard lines.

B. SDPN Applications

The SDPN framework allows for extensibility via applications that comprise the application layer. This allows for integration with other architectures or services and the ability to evolve the SDPN to meet yet undefined needs. Five novel SDPN applications are defined here as a core subset of required applications to enable SDPN functionality.

1) Trust Anchor: A trusted third-party (such as a Certificate Authority, Internet Service Provider, or Regional Internet Registry) must operate as an *endorsing authority* to issue a root certificate to the SDPN controller [4]. This establishes the first SDPN controller as the SDPN trust anchor. The trust anchor application enables trust operations needed to maintain, extend, revoke, and evaluate trust throughout the SDPN. This may include functions such as issuing ephemeral or intermediate certificates, key rotation, or trust evaluation. All network traffic within an SDPN must be encrypted and secured with approved cryptographic protocols and key material derived from this trust anchor.

2) Trust Table: To make Zero Trust-based decisions, the concept of a trust algorithm is introduced in [5] to inform judgment on allowing or denying resource access requests. Many data points are evaluated together to comprise a trust evaluation process [19]. To facilitate these judgments in the SDPN, implementing a trust table [4] is needed to maintain trust scores and associated policies for communication within and between an SDPN. This enables the SDPN controller to operate as the Policy Decision Point (PDP) for the SDPN.

The trust table facilitates cross-referencing of several data points to provide risk-based decision scores on a granular, per-transaction basis. As an example, a service request from HostNodeA to ServiceNodeB may have a different authorization (e.g., allow HTTPS) than an alternate request from

HostNodeA to ServiceNodeB for a different service (e.g., deny telnet).

3) Threat-informed Name Resolution: A significant modification over a traditional SDP workflow is an integration with name resolution, such as DNS. While DNS is arguably the most common name resolution protocol, in SDPN we abstract DNS to "name resolution" as future methods of resolving domains to IP addresses may be used within and between SDPNs based on alternative conventions.

When a host node sends a new name resolution query to its switch perimeter node, the node will follow traditional SDN logic and forward the request up to the SDPN controller. The controller will match the request against the trust table to evaluate if the source and destination network flow is allowed. If the source and destination pair is authorized, the controller will then attempt to resolve the name resolution query.

Security is added to the SDPN by denying name resolution to addresses that have a negative reputation based on cyber threat intelligence (CTI) reporting. CTI feeds can be maintained by a separate application. If the name resolution query escalates beyond the SDPN into a traditional recursive DNS query, DNS replies are subject to various attacks [26]. To prevent resolution to known-bad addresses, the threat-informed name resolution application will match all DNS replies against CTI and will deny resolution to known bad IP addresses.

Name resolution combined with flow management in an SDPN provides two security mechanisms for nodes. First, it will not allow resolution to IPs that are either prohibited by policy or a malicious IP address. Second, if a node were to attempt to directly access an IP address without name resolution, an attempt to communicate with any IP address will ultimately reach the SDPN controller for a trust decision on whether to allow the network flow to be created. If the desired IP address is reported as a bad IP address, the flow will not be created.

4) Network Flow Management and SPA: By introducing the SDPN Controller into the name resolution process, it can manage and trigger traditional SDP and SDN processes upon name resolution. To illustrate, before sending a name resolution response to the requesting host, the controller will notify the service node to expect an incoming SPA from the host node. After this, the controller will send an add flow message to the relevant switch nodes. This will update the switch node flow table to allow the host node to communicate with the service node. After these two control communications have taken place, the controller will then send the name resolution to the requesting host node.

5) SDPN Peering: With trust as a fundamental component of an SDPN, trusted peering can be introduced. Specifically, various trust operations (such as mutual authentication) can take place between SDPNs to provide security assurances and protections before any node-to-node communication traverses a perimeter. Trusted peering can also facilitate SPA exchanges, trusted peering, spoof-resistant communication, and trusted routing.

C. SDPN Management Protocols

To facilitate dynamic network elasticity, SDPN protocols are needed to enable scaling and trusted peering. SDPN protocols define the SDPN controller messages that are needed within and between SDPNs. This includes all the functionality needed for internal management of an individual SPDN, such as joining and leaving the SDPN, key management, status checks, and authentication messages. SDPN messages traverse the control channel between the SDPN controller and nodes. This section defines three initial high-level protocols:

- SDPN Join: This message is a request to add a node to the SDPN. If the SDPN controller is open to onboarding nodes, a sequence of challenge-response trust exchanges, device health checks, and route telemetry evaluation will ensue to provide assurance that the requesting node is legitimate. Parameters such as MAC address, network distance, operating system, can comprise a trust score for the controller to evaluate accepting the node. The SDPN controller is the authority on what nodes may be granted access to the perimeter and at what level of permissions.
- SDPN Leave: This message is sent from a node to the SDPN controller whenever an authenticated node wants to leave the SDPN. This will trigger a sequence of events that will revoke any key material from the node, adjust network flows, and update the SDPN trust table accordingly.
- SDPN Peering: This is a suite of SDPN protocols that contain all the inter-SDPN messages needed to exchange and maintain external trust and routing information. Sample protocols include BGP and OSPF.

IV. SDPN in Operation

A. Building an SDPN

To build an SDPN, the initial SDPN controller must be instantiated as the trust anchor by obtaining a root certificate from a trusted third-party. This is a fundamental requirement of the SDPN. As the trust anchor, it can build, maintain, extend, and revoke trust within the SDPN. Next, a perimeter node must join the SDPN in order to route network packets. The first perimeter node must be locally connected to its associated SDPN controller. Any node greater than one network hop away will not be allowed to initially authenticate. This is to prevent tampering from untrusted network hops between the initial controller and first perimeter node. This network link may be a physical or virtual implementation (e.g., the controller and perimeter node may reside on the same hypervisor). Once an SDPN controller and perimeter node are instantiated, subsequent nodes can join and leave the SDPN at-will.

B. End-to-end Sequence

The sequence of message exchanges that follow are based upon the logical SDPN architecture depicted in Fig. 1. To clearly differentiate SDPN against a reference SDP implementation, it is beneficial to review the process flow defined for SDP 2.0 [22]. In a "traditional" SDP implementation – i.e.,

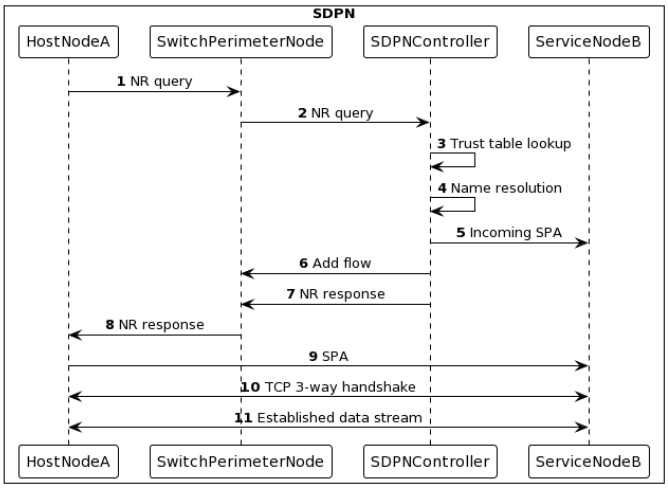

Fig. 2. SDPN message sequence for HostNodeA to communicate with ServerNodeB.

not an SDPN – if HostA wants to communicate with HostB, it will authenticate with the SDP via the following high-level process [22]:

1) HostA (the Initiating Host, or IH) authenticates with the SDP Controller.
2) Upon successful authentication, the SDP Controller determines all Accepting Hosts (AHs) that HostA is authorized to communicate with.
3) SDP Controller informs all authorized AHs to expect an incoming SPA from HostA.
4) SDP Controller replies to HostA with a list of all authorized AHs.
5) HostA sends a SPA packet to every AH, followed by a TCP 3-way handshake.

Through this exchange, there is a contradiction of Zero Trust principles in that upon IH authentication, it is then given a full list of all authorized AHs. If the IH were to become compromised at any point *after* authentication, it now has the ability for an unauthorized user to access all AHs that have been needlessly disclosed.

SDPN modifies this traditional SDP process to include name resolution as discussed in Section III-B3. This pushes security control earlier into the network sequence before any packets are even crafted for the destination node. The entirety of this process is depicted in Fig. 2.

The process is as follows: HostNodeA will send a name resolution query to the SwitchPerimeterNode, which will be forwarded up to the SDPN Controller. Escalating to the controller is the default behavior when a switch perimeter node does not have a corresponding flow for an unknown request.

The SDPN Controller first performs a lookup on the trust table and confirms that HostNodeA is authorized to communicate with the requested hostname (i.e., ServiceNodeB). If authorized, the SDPN Controller will perform the name service resolution based on policy and configuration. Assuming the request resolves to a destination internal to the SDPN, the controller will trigger a series of messages to facilitate the connection before sending the name service response.

First, it will alert ServiceNodeB that an authorized communication from HostNodeA is forthcoming and to expect an incoming SPA from it. Next, it will direct the SwitchPerimeterNode to add a network flow from HostNodeA to ServiceNodeB. This aids in network efficiency as it establishes the network flow ahead of being built via traditional SDN workflows, effectively reducing one round trip SwitchPerimeterNode and controller message exchange. Lastly, the SDPN Controller will provide the name resolution response to the requesting node, HostNodeA. HostNodeA will send a SPA to ServiceNodeB. ServiceNodeB will validate the SPA packet and the TCP 3-way handshake will follow, and communication between HostNodeA and ServiceNodeB can occur.

V. Discussion

The Software-defined Perimeter Network (SDPN) model defined in this paper provides a reference architecture for integrating SDP and SDN. The SDPN is an abstracted method to manage, secure, and monitor network functions and system nodes. The SDPN provides scalable management policies that benefits all devices within the virtual perimeter.

Prior work has also proposed SDN-SDP integrations, but with different goals [27] than the integrated SDPN framework presented here. [28] placed SDP atop SDN with the goal of leveraging SDP to protect services on an SDN network. This proposed integration in [28] manifested into an acknowledgment of the SDN-SDP integration potential in a Cloud Security Alliance working paper [29] and SDP Specification 2.0 [22]. [30] introduces an SDN-SDP integration with a one-time key protocol for securing the paths between the data and control abstraction layers. Lastly, [31] illustrates the educational opportunities afforded by a localized SDN-SDP test bed.

Our model builds upon prior work by clearly defining an integrated reference framework in alignment with three abstraction layers of data, controller, and application planes and an alignment with Zero Trust tenets. While these prior works illustrate SDP built upon an SDN, they do not propose an integration where SDP and SDN controller logic and network flows are tightly coupled. Lastly, our model introduces a trust anchor in the SDPN, which can enable future functionality.

In order to contextualize SDPN future directions, it is worth a higher-level discussion of present day network perimeters. At the highest level of the Internet, a digital perimeter is established as Autonomous System (AS) and is identifiable via an Autonomous System Number (ASN). An ASN is a uniquely assigned number that publicly defines the perimeter and associated sovereignty established by that edge router. Autonomous Systems exchange and maintain peering information via Border Gateway Protocol (BGP). BGP is a cleartext protocol and thus subject to attacks such as redirection, hijacking, and tampering [32], [33]. Various BGP security enhancements have been proposed but a universal solution has not been adopted [34].

To address these challenges, SDPN can tightly couple all traditional network communication with identity, trust, and authorization. With the trust anchor established as a foundational element, each SDPN can be cryptographically identified. With cryptographic protections, we can introduce inter-perimeter trust operations, trusted route selection, and route provenance. This in turn can enable trusted, anti-spoofable, and tamper-resistant communication. SDPN could provide a viable approach for securing inter-BGP communications.

VI. Conclusions

This paper introduced the Software-defined Perimeter Network (SDPN), an integration of Software-defined Perimeter (SDP) and Software-defined Networking (SDN) into a single virtualized network framework. The SDPN framework aligns with existing SDP, SDN, and Zero Trust models by maintaining separation of network and perimeter functionality into data, control, and application planes. Through introducing trust at the SDPN controller as a trust anchor, the SDPN framework provides a viable model for future network security paradigms.

References

[1] J. L. Bayuk and B. M. Horowitz, "An architectural systems engineering methodology for addressing cyber security," *Systems Engineering*, vol. 14, no. 3, pp. 294–304, Sep. 2011. [Online]. Available: http://doi.wiley.com/10.1002/sys.20182

[2] R. Savold, N. Dagher, P. Frazier, and D. McCallam, "Architecting cyber defense: A survey of the leading cyber reference architectures and frameworks," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2017, pp. 127–138.

[3] J. Garbis and J. W. Chapman, *Zero Trust Security: An Enterprise Guide*. Berkeley, CA: Apress, 2021.

[4] M. Bursell, *Trust in computer systems and the Cloud*. Hoboken, NJ: John Wiley & Sons, Inc., 2022.

[5] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," National Institute of Standards and Technology, Tech. Rep., August 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

[6] N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN: An Intellectual History of Programmable Networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, Apr 2014.

[7] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 1–12, Aug. 2007. [Online]. Available: https://doi.org/10.1145/1282427.1282382

[8] M. Casado, M. Freedman, J. Pettit, Jianying Luo, N. Gude, N. McKeown, and S. Shenker, "Rethinking Enterprise Network Control," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1270–1283, 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5169973/

[9] M. Casado, N. McKeown, and S. Shenker, "From ethane to sdn and beyond," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 5, p. 92–95, Nov 2019. [Online]. Available: https://doi.org/10.1145/3371934.3371963

[10] A. A. Barakabitze and R. Walshe, "SDN and NFV for QoE-driven multimedia services delivery: The road towards 6G and beyond networks," *Computer Networks*, vol. 214, p. 109133, Sep. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1389128622002523

[11] M. Carver, L. W. DiValentin, M. L. Lefebvre, E. Hovor, and D. W. Rozmiarek, "Method and system for automated incident response," U.S. Patent 10,051,010, Aug. 14, 2018.

[12] L. W. DiValentin, M. Carver, M. L. Lefebvre, D. W. Rozmiarek, and E. Ellett, "Deception network system," U.S. Patent 10,447,733, Oct. 15, 2019.

[13] M. Ojo, D. Adami, and S. Giordano, "A sdn-iot architecture with nfv implementation," in *2016 IEEE Globecom Workshops (GC Wkshps)*, 2016, pp. 1–6.

[14] X. Duan, Y. Liu, and X. Wang, "Sdn enabled 5g-vanet: Adaptive vehicle clustering and beamformed transmission for aggregated traffic," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 120–127, 2017.

[15] J. C. Correa Chica, J. C. Imbachi, and J. F. Botero Vega, "Security in SDN: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, p. 102595, Jun. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1084804520300692

[16] R. Deb and S. Roy, "A comprehensive survey of vulnerability and information security in SDN," *Computer Networks*, vol. 206, p. 108802, Apr. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1389128622000299

[17] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash, and M. Shaheed, "SDN Security Review: Threat Taxonomy, Implications, and Open Challenges," *IEEE Access*, vol. 10, pp. 45 820–45 854, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9760465/

[18] J. Kindervag, S. Balaouras *et al.*, "No more chewy centers: Introducing the zero trust model of information security," *Forrester Research*, vol. 3, 2010.

[19] D. Horne and S. Nair, "Introducing zero trust by design: Principles and practice beyond the zero trust hype," in *Advances in Security, Networks, and Internet of Things*. Springer, 2021, pp. 512–525.

[20] "Executive order on improving the nation's cybersecurity," *Executive Order No. 14028*, 2021. [Online]. Available: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[21] Software Defined Perimeter Working Group, "Sdp specification 1.0," Cloud Security Alliance (CSA), April 2014.

[22] ——, "Software-defined perimeter (sdp) specification v2.0," Cloud Security Alliance (CSA), March 2022.

[23] "Single Packet Authorization: A Comprehensive Guide to Strong Service Concealment with fwknop." [Online]. Available: http://www.cipherdyne.org/fwknop/docs/fwknop-tutorial.html

[24] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, p. 101–118, 2018. [Online]. Available: https://doi.org/10.1016/j.jnca.2017.11.015

[25] K. ElDefrawy and T. Kaczmarek, "Byzantine Fault Tolerant Software-Defined Networking (SDN) Controllers," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. Atlanta, GA, USA: IEEE, Jun. 2016, pp. 208–213. [Online]. Available: http://ieeexplore.ieee.org/document/7552205/

[26] T. H. Kim and D. Reeves, "A survey of domain name system vulnerabilities and attacks," *Journal of Surveillance, Security and Safety*, vol. 1, no. 1, pp. 34–60, 2020.

[27] L. Wang, H. Ma, Y. Jiang, Y. Tang, S. Zu, and T. Hu, "A data plane security model of segmented routing based on SDP trust enhancement architecture," *Scientific Reports*, vol. 12, no. 1, p. 8762, 2022. [Online]. Available: https://www.nature.com/articles/s41598-022-12858-2

[28] A. Sallam, A. Refaey, and A. Shami, "On the security of sdn: A completed secure and scalable framework using the software-defined perimeter," *IEEE Access*, vol. 7, pp. 146 577–146 587, 2019.

[29] J. Koilpollai and N. A. Murray, "Software defined perimeter (sdp) and zero trust," Cloud Security Alliance (CSA), May 2020.

[30] T. W. Kim, Y. Pan, and J. H. Park, "Otp-based software-defined cloud architecture for secure dynamic routing," *CMC-COMPUTERS MATERIALS & CONTINUA*, vol. 71, no. 1, pp. 1035–1049, 2022.

[31] D. Home, "Leveraging software defined perimeter (sdp), software defined networking (sdn), and virtualization to build a zero trust testbed with limited resources," in *Advances in Security, Networks, and Internet of Things*. Springer, 2022.

[32] O. Nordström and C. Dovrolis, "Beware of bgp attacks," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, p. 1–8, Apr 2004. [Online]. Available: https://doi.org/10.1145/997150.997152

[33] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of bgp security issues and solutions," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, 2009.

[34] A. Mitseva, A. Panchenko, and T. Engel, "The state of affairs in BGP security: A survey of attacks and defenses," *Computer Communications*, vol. 124, pp. 45–60, Jun. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S014036641731068X