

Security of software defined networks: evolution and challenges

Siham Aouad¹, Issam El Meghrouni², Yassine Sabri³, Adil Hilmani², Abderrahim Maizate²

¹Smart Systems Laboratory, National School of Computer Science and Systems Analysis, Mohamed V University, Rabat, Morocco

²RITM-ESTC/CED-ENSEM, Hassan II University, Casablanca, Morocco

³Laboratory of Innovation in Management and Engineering for Enterprise (LIMIE), ISGA Institut Supérieur d'Ingénierie & des Affaires, Rabat, Morocco

Article Info

Article history:

Received Mar 4, 2023

Revised Apr 18, 2023

Accepted May 8, 2023

Keywords:

Intrusion detection systems

Intrusion prevention system

OpenFlow

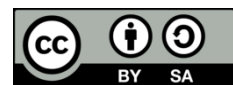
Security

Software-defined networking

ABSTRACT

In software-defined networking (SDN), network traffic is managed by software controllers or application programming interfaces (APIs) rather than hardware components. It differs from traditional networks, which use switches and routers to control traffic. Using SDN, you can create and control virtual networks or traditional hardware networks. Furthermore, OpenFlow allows network administrators to control exact network behavior through centralized control of packet forwarding. For these reasons, SDN has advantages over certain security issues, unlike traditional networks. However, most of the existing vulnerabilities and security threats in the traditional network also impact the SDN network. This document presents the attacks targeting the SDN network and the solutions that protect against these attacks. In addition, we introduce a variety of SDN security controls, such as intrusion detection systems (IDS)/intrusion prevention system (IPS), and firewalls. Towards the end, we outline a conclusion and perspectives.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Siham Aouad

Smart Systems Laboratory, National School of Computer Science and Systems Analysis

Mohamed V University

Avenue Mohammed Ben Abdallah Regragui, Madinat Al Irfane, BP 713, Agdal Rabat, Morocco

Email: siham.aouad@ensias.um5.ac.ma

1. INTRODUCTION

In the last several years, traditional computer networks have experienced great challenges that mainly result from modern applications deployed on these networks. The paradigm of software-defined networking (SDN) emerged within this particular context, this enables us primarily to adjust to the constantly changing nature of various applications. Indeed, SDN mainly allows the centralization of the logic determining the management policies of a network in one or more units called controllers. These controllers communicate with the rest of the network equipment. Therefore, defining a network management policy amounts to writing programs and deploying them in the controllers. In most cases, these programs will be compiled, taking into account the topology and the available resources, to generate the configurations necessary for each network device to implement the desired policies. Several prominent companies and universities, including Google and Stanford, have embraced the SDN architecture. On the other hand, manufacturers of network equipment such as Cisco, HP, and Juniper now offer SDN solutions that make it possible to manage data centers.

SDN, an inventive architecture, enables the separation of the control plane from the data plane, allowing them to evolve and develop independently. Ethane [1] was one of the first protocols used in SDN. Using a centralized control architecture, this model was suggested for corporate networks. In order to support the Ethane protocol, custom switches were needed (NetFPGA, OpenWrt, Linux). SDN's main protocol is

Openflow [2], an open architecture originally developed to allow researchers to experiment on heterogeneous networks without impacting actual traffic. In the switch data plane, Openflow [3] aims to provide more configuration options. Openflow involves communication between the data plane and the centralized control plane, allowing for the network to be entirely managed through user software applications application programming interfaces (APIs). With the implementation of OpenFlow [3], the restrictions of static protocols are eliminated, facilitating quick advancement and stimulating exploration of new network technologies. The subsequent sections of the paper are structured in the following manner: in section 2, a definition of the SDN architecture is presented. Then, section 3 presents security attacks and solutions for SDN. Subsequently, section 4 explores the challenges associated with SDN. In conclusion, we provide our perspectives on this work.

2. ARCHITECTURE OF SOFTWARE-DEFINED NETWORKING

By utilizing software applications, SDN enables intelligent control over the network's functioning, making it a distinctive network architecture approach [4]. The traditional network environment that relies on predetermined software and hardware from network hardware vendors undergoes a transformation through SDN, into a flexible, centrally managed software and management environment, enabling network abstraction programming. Then, it is based on a structure with three superimposed planes: a data plane, a control plane, and an application plane. The southbound interface (SBI) at the SDN switch level facilitates communication, while the northbound interface (NBI) located in the control plane ensures communication between controllers and applications [5].

2.1. Data plan

An SDN infrastructure includes network equipment (switches, routers, and middleboxes) similar to a traditional network. Physical devices have become simple transmission elements without integrated controls or software for autonomous operation. Transmission of data flow occurs through the data plane. Data plane devices provide intelligence to the controller, a logically centralized system. Its main task is to forward incoming flows to their destinations, using routes defined in flow tables. Furthermore, these novel networks are constructed using open and standard interfaces like OpenFlow, which ensure interoperability, communication compatibility, and configuration. As shown in Figure 1, SDN/OpenFlow architectures consist of two main components: controllers and forwarding devices.

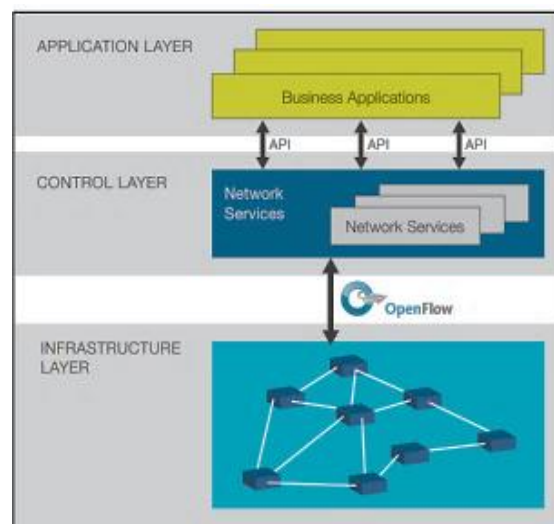


Figure 1. Architecture of SDN [6]

2.2. Control plane

The network operating system (NOS) in SDN logically centralizes network control, simplifying network management and issue resolution [7]. Similar to traditional operating systems, a NOS offers developers abstractions, vital services, and APIs. The NOS can offer generic capabilities as services, including network status and topology information, network configuration distribution, and device discovery. Thanks to the NOS, a developer is relieved from the burden of handling the intricacies of data distribution

among routing elements. By reducing the complexity associated with developing network protocols and applications, such systems may allow innovation to occur at a faster rate.

2.3. Application plane

As shown in Figure 1, the application layer is positioned above the control layer. By means of the northbound interface, SDN applications can obtain a real-time global perspective of the network, including its instantaneous status, as exemplified by application layer traffic optimization (ALTO) [8]. Armed with this information, SDN applications can leverage a high-level language offered by the control layer to devise tactics for controlling the underlying physical networks. SDN provides the "Platform as a Service" paradigm for networking in this field [9].

2.4. Southbound API

The southbound API, which bridges forwarding devices and control planes, is among the most crucial elements of an SDN system. The southbound API allows the controller to control network behavior by managing flow inputs from all the underlying switches, in addition to other functionalities, protocol plugins (such as simple network management protocol (SNMP), border gateway protocol (BGP), and NetConf) enable the controller to oversee both new and existing physical or virtual devices through southbound APIs (like OpenFlow, POF, OpFlex, and OpenState). At present, OpenFlow is widely considered as the standard for the SBI in SDN [10].

2.5. Northbound

The northbound interface (NBI) is the interface between applications and the controller. Currently, OpenFlow is widely accepted as the southbound API, but a common northbound API is yet to be developed. As use cases are still being developed, it might be too early to define a standard northbound API [11]. Either way, expect a common northbound API to emerge as SDN evolves. Taking advantage of SDN's full potential requires an abstraction that allows applications to be independent of specific implementations.

3. SECURITY ATTACKS AND SOLUTIONS FOR SDN

3.1. Security attacks in the SDN network

SDN is a type of network architecture that involves the separation of the data plane and the control plane, enabling more efficient and flexible network management. However, as with any network architecture, SDN networks are susceptible to security attacks. One common type of attack on SDN networks is known as the distributed denial of service (DDoS) attack, which floods the network with traffic, causing it to slow down or crash. Additionally, SDN networks that rely on centralized controllers may be more vulnerable to attacks on the controller, which can have a ripple effect on the entire network. SDN architecture offers several advantages, yet it suffers from security issues:

- Open programmable APIs: APIs with open programming interfaces may cause security issues when attackers exploit them via software security vulnerabilities like injections of malicious code or web attacks like cross-site scripting (XSS).
- Controller issues: controllers are centralized, making them single points of failure and attack targets; therefore, they can be a target for attackers.
- SDN switches issues: SDN switches are susceptible to DDoS attacks, in which an attacker can flood the flow table with numerous large packets, due to the restricted number of flow tables [12].

After conducting the analysis, it was found that both traditional networks and SDN are susceptible to security threats such as tampering, spoofing, DoS, and information disclosure. To mitigate such attacks, it is crucial to implement robust security measures including access control, traffic filtering, and encryption. Additionally, it is essential to keep the network devices and software updated and patched on a regular basis to address any known vulnerabilities.

3.1.1. Application layer attacks

Application layer attacks in SDN are a type of security threat that targets the application layer of the SDN architecture. Application layer attacks aim to exploit vulnerabilities in the network applications running on the SDN infrastructure. These vulnerabilities can be caused by errors in the software code, misconfiguration, or insufficient security measures. Above we quote some control layer attacks:

- Unauthorized access: the SDN application layer is located on the control layer that encompasses the tangible aspects of SDN functionalities. Generally, applications are not developed by controller vendors, but by third parties, who do not take security concerns into account during the development

process [13]. Thus, attackers can exploit this and gain unauthorized access to sensitive information and make changes without having to authenticate [14].

- Malicious code: an attacker could potentially inject malicious code into the network, either through the SDN controller or through one of the forwarding devices. This code could then be executed on the network, potentially allowing the attacker to gain access to systems or steal sensitive information.
- Policy insertion: different fields such as cloud computing use SDN extensively with a variety of services and sophisticated applications. Consequently, security rules can conflict when created and inserted.

3.1.2. Control layer attacks

Control layer attacks in SDN refer to attacks that target the control plane of the SDN architecture. Control layer attacks aim to disrupt the operation of the control plane, either by modifying the control messages or by injecting fake control messages. We quote some control layer attacks:

- Denial of service and distribute: attackers impose floods of requests on the controller by flooding the network with flooded requests that saturate the controller by consuming its resources. These attacks can target the channel level, the controllers, or the controllers and switches. SDN controllers are challenged by DDoS/DoS attacks that disrupt legitimate users' access to the network [13], [15].
- Attacks from applications: as mentioned before, the application layer sits above the control layer. By getting unauthorized access to the application layer, an attacker can obtain data over the network and attack the control layer.
- Attacks that target distributed multi-controllers distributed multi-controllers divide the network into sub-networks to handle the growing network range. Additionally, this solution can create security problems, such as implementing an efficient security policy and managing configuration conflicts [16].

3.1.3. Infrastructure layer attacks

Infrastructure layer attacks in SDN are a type of security threat that targets the physical components of the SDN infrastructure. Infrastructure layer attacks aim to exploit vulnerabilities in the network infrastructure by targeting network devices such as switches and routers. We quote some control layer attacks:

- DoS attack: in order to intercept the flow table and buffer flow, for an attacker to generate new rules for the flow table, they must send large and unfamiliar packets continuously. As a result of this attack, legitimate packets are dropped and compromised [16], [17].
- Man-in-the-middle: as a man-in-the-middle attack, the attacker intercepts information flowing between the switches and the controllers without being detected. As a result of this attack, additional attacks that can be executed include black hole attacks and eavesdropping [13], [17].
- Switch spoofing attacks: these attacks involve a malicious actor posing as a legitimate switch in the SDN network in order to gain unauthorized access to the network or to steal sensitive information.
- Flow-table overflow attacks: these attacks involve overwhelming the flow table of an SDN switch with a large number of flow table entries, causing the switch to become unresponsive or malfunction.

3.2. Proposed SDN security solutions

As mentioned in section 3.1, threats and requirements differ at each layer of the SDN architecture. These requirements need to be met in order to prevent various types of security threats and attacks. Layers can be affected by communication flood attacks between the controller and the switch. Unauthorized access to the controller can be a consequence of authorization attacks. The following are the suggested solutions to the major security issues in SDN:

- Use of transport layer security (TLS): one of the most well-known security protocols today is TLS. Several network-based applications and services use the TLS protocol to secure end-to-end communications, including web browsers and server software, email clients and servers, instant messengers, and teleconference applications. TLS's security has been closely scrutinized, and it has been updated over time to maintain high-security levels. The TLS protocol is used to secure north-south communication. It prevents flow injection by encrypting flows [18]. The use of TLS in SDN provides several benefits, including confidentiality, integrity, and authentication. Confidentiality is achieved by encrypting the communication between the SDN components, ensuring that the information exchanged cannot be intercepted and read by unauthorized parties [19]. Securing the communication between switches and the controller is a crucial use case of TLS in SDN. The importance of this is that the controller is responsible for overseeing the switches' operation. If the communication between them is compromised, an attacker may gain control of the network. Securing the communication between the SDN controller and various network components, including the OpenFlow switch, is another use case of

TLS in SDN. By using TLS, the controller can ensure that it is communicating with the intended switch and that the communication is encrypted and secure [20]. In conclusion, the use of TLS in SDN is important for securing communication between different SDN components. It provides confidentiality, integrity, and authentication, ensuring that the communication is secure and cannot be intercepted or tampered with by unauthorized parties.

- Firewall implementation: traditionally, firewalls are deployed as distinct devices from network devices. Forwarding and dropping packets between network devices that support OpenFlow are simple to implement in SDN networks since it is among the features of the protocol implying that establishing distributed firewall operations on all OpenFlow-supported network devices is straightforward. Several tasks that are usually managed by firewalls are executed by SDN controllers. For example, Flow rules are written into switches' flow tables by controllers in SDN to determine the fate of flows. Controllers keep rules or access control lists (ACLs) for every switch in an SDN network. Traditional networks do not support such connections (between firewalls and switches). OpenFlow has extended flow attributes in recent versions, letting SDN-based firewalls handle flow and packet attributes more specifically.
- Intrusion detection/protection systems (IDS/IPS): by comparing packet signatures with existing threat inventories, mining data, and recognizing patterns, intrusion detection and prevention systems (IDS/IPS) are able to allow or stop packets [21].
- Network scan detection: one approach to detecting attached connections is the threshold random walk method [22], which examines the outcome of TCP connections. There are statistically more failures than successes, but still enough differences to distinguish TCP packets into the failure of each connection. With SDN, flows can be monitored by sessions by controlling the flow rules. In these protocols, the amount of packets forwarded to the server is used to verify the scan, since no session information is available. The TCP protocol, however, is generally used for most network scanning attacks.

4. SECURITY CHALLENGES WITH SDN

SDN paradigm offers several advantages, such as flexibility, agility, and scalability. Furthermore, it presents numerous security challenges that must be tackled to safeguard the network's security. The following are the significant security challenges linked with SDN.

- Centralization: SDN's centralization of the control plane is one of the primary security challenges, as one potential drawback is that it creates a single point of failure and a possible target for attacks. An attacker who gains control of the central controller can compromise the entire network [23].
- Communication security: another significant security challenge in SDN is securing the communication between the controller and switches. In the event of this communication being compromised, it would allow an attacker to manipulate the network traffic, leading to data theft or unauthorized access [24].
- Malicious applications: SDN allows for the creation and installation of applications on the controller, which can access the network infrastructure. If a malicious application is installed, it can present a substantial risk to the network's security [25].
- Authorization and access control: in SDN, the controller has complete control over the network, which means that proper authorization and access control mechanisms must be in place to prevent unauthorized access and manipulation of the network [26].
- Denial of service attacks: since the control plane in SDN is centralized, it is susceptible to DoS attacks. Attackers can flood the controller with a high volume of requests, causing it to become unresponsive and preventing it from controlling the network.

SDN solutions face a number of security challenges, among them robustness and scalability. Conducting stateful inspections at advanced levels or on numerous subjects necessitates substantial memory, storage, and network resources. Managing the surge in the number of states poses a challenge as any modification in the traffic flows or network elements alters the network state, which is defined by rules and the traffic flows within the network. There can be a huge number of possible states as a result. The utilization of FlowTest for the evaluation of SDN policies and firewall scenarios in stateful network scenarios was suggested by Casado *et al.* [27]. In order to test stateful behaviors systematically, they emphasized data plane testing. In general, policies include high-level instructions that are stateful. "Block unsolicited Internet connections" is an example of a policy. There is no reference to any (L2-L3) information in such a policy. Firewalls must work beyond the L2-L3 layer in order to accomplish this. There are different states of TCP connections (e.g., new, null, ... invalid, or established). Rather than representing network states, they represent traffic states. Moreover, a session-level proxy module is also proposed to support stateful inspection. Hypertext transfer protocol (HTTP) objects are used to express proxy state. In addition, providing access control solutions that are flexible and

dynamic is the goal of SDN. A fine-grained access control policy is provided by the Ethane SDN architecture [28]. SDN was inspired by the Ethane protocol and the OpenFlow protocol, which enables central control over networks. A central controller and flow-based networks are used in Ethane. Makes decisions about flows by switching them directly to the controller. The central controller maintains policies. Therefore, the primary objective is to ensure the security of the controller and its communication. According to Huang *et al.* [29] PermOF is a fine-grained system for SDN access control. The PermOF provides 18 levels of permission to minimize the possibility of privilege escalation or intrusion. As part of the permission system, run-time isolation (between the controller and applications) is also implemented. An OpenFlow application is granted the least privilege by default. The applicability of such approaches depends on the ability to isolate applications from controllers. On the other hand, there are also other challenges in the SDN network which concern traffic monitoring tools and intrusion detection. Traffic monitoring can be implemented using dynamic measurements aware routing or forwarding [30]. In the discussion, three challenges were addressed: dynamic evaluation of traffic importance, flow aggregation and minimizing network disruptions. These tasks are based on information from OpenFlow switches. A flow's importance is determined by its size. The flow-query and flow-expire messages allow controllers to determine the size of flows. In terms of intrusion detection, Snort integration with SDN poses several challenges. It has been proposed in [31] to integrate Snort with OpenFlow networks. As a result of SnortFlow's ability to detect and prevent intrusions in real-time, the cloud system can be reconfigured on the spot. Three components make up SnortFlow: a daemon that collects alerts, an alert interpreter that parses alerts and determines which traffic to focus on, and a generator of rules that produces OpenFlow rules. As a result of the new rules, changes are saved so they can be rolled back or restored if needed.

5. CONCLUSION

With the advent of software-defined networks, the requirement for a well-managed, trustworthy, flexible, and secure network has been met. In contrast to traditional networks, SDNs are more vulnerable to attacks because of the separation of the two planes. As a result, network and control traffic could be compromised rigorously in terms of availability, consistency, authenticity, confidentiality, and integrity. This paper provides a concise summary of the current research on security threats in SDNs, as well as security controls. As SDN research and development progresses, the security landscape changes in SDN. It may be necessary to take specific measures to counteract security threats associated with SDN protocols and APIs. We end by considering various SDN security concerns and research areas. The privileges of insider threats in SDN are often higher, particularly if they can gain entry to the controller modules or resources. Security attacks can propagate within virtualized SDN environments either intentionally or unintentionally. Continuously assessing security measures is crucial to guarantee the complete isolation of tenants who share the same physical network. Another type of insider threat is the compromise of controller resources. Controller APIs can be exploited as backdoors by applications interacting with the controller. This type of attack can cause severe network damage because the controller has such privileges. Another research topic is the policy life cycle; The SDN offers the possibility of implementing policy lifecycle activities automatically. Managing policy requires the integration of low-level mechanisms with high-level requirements, which presents a major challenge. For traditional networks, firewall rules at low levels need to be contextualized. The third topic is application access control; There is no practical way to completely deny the host when a security attack compromises certain applications. Users/hosts and switches/networks can be targeted for access control with SDN global policies. The controller can create a central access control module to oversee information from various levels and allow or block traffic depending on the information. In this way, access control information can be changed dynamically.




REFERENCES

- [1] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," *Computer Communication Review*, vol. 37, no. 4, pp. 1–12, 2007, doi: 10.1145/1282427.1282382.
- [2] K. Greene, "MIT Tech Review 10 breakthrough technologies: software-defined networking," 2009, Accessed: May 05, 2023. [Online]. Available: <http://www.technologyreview.com/article/412194/tr10-software-defined-networking/>.
- [3] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.
- [4] "OpenFlow switch specification 1.4.0," Opennetworking, 2013. Accessed: May 05, 2023. [Online]. Available: <https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf>.
- [5] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *Journal of Network and Computer Applications*, vol. 67, pp. 1–25, May 2016, doi: 10.1016/j.jnca.2016.03.016.
- [6] N. Gude, J. Pettit, B. Pfaff, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 3, pp. 105–110, 2008, doi: 10.1145/1384609.1384625.
- [7] V. K. Gurbani, M. Scharf, T. V. Lakshman, V. Hilt, and E. Marocco, "Abstracting network state in software defined networks (SDN) for rendezvous services," in *2012 IEEE International Conference on Communications (ICC)*, Jun. 2012, pp. 6627–6632, doi: 10.1109/ICC.2012.6364858.




- [8] E. Keller and J. Rexford, "The 'platform as a service' model for networking," in *2010 Internet Network Management Workshop / Workshop on Research on Enterprise Networking, INM/WREN 2010*, 2010.
- [9] H. Jamjoom, D. Williams, and U. Sharma, "Don't call them middleboxes, call them middlepipes," in *Proceedings of the third workshop on Hot topics in software defined networking*, Aug. 2014, pp. 19–24, doi: 10.1145/2620728.2620760.
- [10] J. Dix, "Clarifying the role of software-defined networking northbound APIs," *Networkworld*, 2012. Accessed: Apr. 30, 2023. [Online]. Available: <http://www.networkworld.com/article/2165901/lan-wan/clarifying-the-role-of-software-defined-networking-northbound-apis.html>.
- [11] M. Arif, G. Wang, and V. E. Balas, "Secure VANETs: Trusted communication scheme between vehicles and infrastructure based on fog computing," *Studies in Informatics and Control*, vol. 27, no. 2, pp. 235–246, Jan. 2018, doi: 10.24846/v27i2y201811.
- [12] Z. Shu, J. Wan, D. Li, J. Lin, A. V. Vasilakos, and M. Imran, "Security in software-defined networking: threats and countermeasures," *Mobile Networks and Applications*, vol. 21, no. 5, pp. 764–776, Oct. 2016, doi: 10.1007/s11036-016-0676-x.
- [13] OWASP Top 10 application security risks," OWASP, 2010, Accessed: Apr. 30, 2023. [Online]. Available: https://www.owasp.org/index.php/Top_10_2010-Main.
- [14] P. Zhang, H. Wang, C. Hu, and C. Lin, "On denial of service attacks in software defined networks," *IEEE Network*, vol. 30, no. 6, pp. 28–33, Nov. 2016, doi: 10.1109/MNET.2016.1600109NM.
- [15] S. Scott-Hayward, "Design and deployment of secure, robust, and resilient SDN controllers," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, Apr. 2015, pp. 1–5, doi: 10.1109/netsoft.2015.7258233.
- [16] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, Aug. 2013, pp. 151–152, doi: 10.1145/2491185.2491222.
- [17] S. E. Schechter, J. Jung, and A. W. Berger, "Fast detection of scanning worm infections," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3224, 2004, pp. 59–81, doi: 10.1007/978-3-540-30143-1_4.
- [18] D. S. Rana, S. A. Dhondiyal, and S. K. Chamoli, "Software defined networking (SDN) challenges, issues and solution," *International journal of computer sciences and engineering*, vol. 7, no. 1, pp. 884–889, 2019.
- [19] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "Sdn Security: A Survey," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov. 2013, pp. 1–7, doi: 10.1109/SDN4FNS.2013.6702553.
- [20] J. Esch, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 10–13, Jan. 2015, doi: 10.1109/JPROC.2014.2374752.
- [21] K. Nam and K. Kim, "A Study on SDN security enhancement using open source IDS/IPS Suricata," in *9th International Conference on Information and Communication Technology Convergence: ICT Convergence Powered by Smart Intelligence, ICTC 2018*, Oct. 2018, pp. 1124–1126, doi: 10.1109/ICTC.2018.8539455.
- [22] S. K. Fayaz and V. Sekar, "Testing stateful and dynamic data planes with FlowTest," in *Proceedings of the third workshop on Hot topics in software defined networking*, Aug. 2014, pp. 79–84, doi: 10.1145/2620728.2620751.
- [23] M. C. Dacier, H. König, R. Cwalinski, F. Kargl and S. Dietrich, "Security challenges and opportunities of software-defined networking," in *IEEE Security & Privacy*, vol. 15, no. 2, pp. 96–100, March-April 2017, doi: 10.1109/MSP.2017.46.
- [24] S. Saraswat, V. Agarwal, H. P. Gupta, R. Mishra, A. Gupta, and T. Dutta, "Challenges and solutions in software defined networking: A survey," *Journal of Network and Computer Applications*, vol. 141, pp. 23–58, 2019, doi: 10.1016/j.jnca.2019.04.020.
- [25] S. Scott-Hayward, S. Natarajan and S. Sezer, "A survey of security in software defined networks," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, Firstquarter 2016, doi: 10.1109/COMST.2015.2453114.
- [26] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, "Securing software defined networks: taxonomy, requirements, and open issues," in *IEEE Communications Magazine*, vol. 53, no. 4, pp. 36–44, April 2015, doi: 10.1109/MCOM.2015.7081073..
- [27] M. Casado *et al.*, "Rethinking enterprise network control," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1270–1283, 2009.
- [28] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for OpenFlow applications," in *HotSDN 2013 - Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Aug. 2013, pp. 171–172, doi: 10.1145/2491185.2491212.
- [29] G. Huang, C. N. Chuah, S. Raza, and S. Seetharaman, "Dynamic measurement-aware routing in practice," *IEEE Network*, vol. 25, no. 3, pp. 29–34, May 2011, doi: 10.1109/MNET.2011.5772058.
- [30] T. Xing, D. Huang, L. Xu, C. J. Chung, and P. Khatkar, "SnortFlow: A OpenFlow-based intrusion prevention system in cloud environment," in *Proceedings - 2013 2nd GENI Research and Educational Experiment Workshop, GREE 2013*, Mar. 2013, pp. 89–92, doi: 10.1109/GREE.2013.25.
- [31] C. J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 4, pp. 198–211, Jul. 2013, doi: 10.1109/TDSC.2013.8.

BIOGRAPHIES OF AUTHORS






Siham Aouad    born in Tangier, Morocco in 1981, holds a Ph.D. in Computer Engineering from Mohammadia School of Engineers EMI in 2014. She obtained her network engineering degree from the National School of Applied Sciences ENSA Tangier in 2005. Currently, she works at the department of communication networks at the National School of Computer Science and Systems Analysis ENSIAS. Her research interests span across various areas including wireless communications, WSN, smart cities, SDN, AI, virtualization, cloud computing, and security. She can be contacted at email: siham.aouad@ensias.um5.ac.ma.






Issam El Meghrouni    is currently a Ph.D. student in the RITM (Networks, IT, Telecommunications and Multimedia) Laboratory at Hassan II University. Recognition gesture, machine learning, and deep learning are among his research interests. He can be contacted at email: magrouni@gmail.com.






Yassine Sabri    was born on October 28, 1984, in Rabat, Morocco. He pursued his Ph.D. in the field of WSN Technology at the Laboratory of Science and Technology. In 2013, he joined the Department of Science and Technology at ISGA Rabat, Morocco, as an Assistant Professor. Yassine's research interests encompass a broad range of topics, including wireless sensor networks, evolutionary computation, internet of things (IoT), and mobile computing. He can be contacted at email: yassine.sabri@isga.ma.



Adil Hilmani    after completing his diploma in Network and Telecommunication engineering from the University of Seville in Spain, he went on to obtain his doctorate in computer engineering from ENSEM in Casablanca-Morocco in 2021. Currently, he serves as a professor at OFPPT in Kénitra, Morocco. His research interests lie in the areas of mobile networks and computing, wireless sensor networks, and embedded systems software for IoT. He can be contacted at email: adilhilmani@gmail.com.



Abderrahim Maizate    after completing his diploma in Network & Telecommunication engineering from the University of Seville in Spain, he went on to obtain his doctorate in computer engineering from ENSEM in Casablanca-Morocco in 2021. Currently, he serves as a professor at OFPPT in Kénitra, Morocco. His research interests lie in the areas of mobile networks and computing, wireless sensor networks, and embedded systems software for IoT. He is a member of IEEE. He can be contacted at email: maizate@hotmail.com.