BCFR: Blockchain-based Controller Against False Flow Rule Injection in SDN

Sarra Boukria
USTHB University
Algiers, Algeria
boukriasarra94@gmail.com

Mohamed Guerroumi
USTHB University
Algiers, Algeria
mguerroumi@usthb.dz

Imed Romdhani
Edinburgh Napier University
United Kingdom, Great Britain
I.Romdhani@napier.ac.uk

***Abstract*—Software Defined Networking (SDN) technology increases the evolution of Internet and network development. SDN, with its logical centralization of controllers and global network overview changes the network's characteristics, on term of flexibility, availability and programmability. However, this development increased the network communication security challenges. To enhance the SDN security, we propose the BCFR solution to avoid false flow rules injection in SDN data layer devices. In this solution, we use the blockchain technology to provide the controller authentication and the integrity of the traffic flow circulated between the controller and the other network elements. This work is implemented using OpenStack platform and Onos controller. The evaluation results show the effectiveness of our proposal.**

***Index Terms*—Software Defined Networking, SDN controller security, Blockchain, OpenStack, Onos, injection, false flow rules.**

I. Introduction

Nowadays, due to the development of the Internet, everything is accessible easily, anytime, from everywhere. Then, with the massive use of new ITs trends such as cloud computing [1], big data [2] and IoT (Internet of Things) [3] [4], which represent large amount of data, the computer networks consolidation and the communications ensuring of these different technologies become complex, expensive and very difficult for managing. Furthermore,traditional IP networks are vertically integrated, so they can't ensure the efficiency, the reliability, the flexibility and the robustness to manage the huge amount of data circulating on the network.

In order to address these challenges, the idea of "programmable networks" has been proposed under the concept of SDN [5], which is an emerging networking paradigm that aims to change the limitations of current network infrastructures. SDN breaks the vertical integration by separating the networks control logic (the control plane) from the underlying routers and switches that forward the traffic (the data plane). Furthermore, with the separation of the control and data planes, network switches become simple forwarding devices, and the network control logic is implemented in a centralized controller, which simplifies policy enforcement and network reconfiguration and evolution [6]. Accordingly, the advantages of SDN technology are attracting great attention from both academia and industry.

The spread of SDNs characteristics, which are fundamental for future networks created new security challenges for companies and network administrators. Indeed, these challenges present new attacks methodologies which aim to change the architecture of this technology, for example, the communication channels between the different layers and the controller. Hence, it is essential to secure this new network approach before it is exploited at a large scale [7] [8].

In this work, we focus on SDN controller security. We provide a mechanism for securing the communication between the controller and the SDN network elements by using the blockchain technology [9] [10] [11]. This approach insures the security, storage and exchange of information without centralized control. The aim is to control and secure the communication of registration of all exchanged operations between the controller and the rest of network components, with accordance, reliability and efficiency. To the best of our knowledge, this is the first work that addresses the problem of false flow rules injection in SDN using blockchain.

The blockchain technology is infallible to attacks, due to its architecture, that allows incremental hashing and storage of replication data on a group of trust nodes. So, how we can integrate it, within the framework of SDN to secure the network?

In order to address this problem, we carried out our work in a progressive way. The major contributions of this paper is the implementation of a security method in SDN architecture. This method will be provided as a service to ensure the protection of our network communication and information between the controller and the various integrated network devices such as switches. The rest of the paper is structured as follows. Section II shows the related work. In Section III, we present the concept of our proposal. Section IV describes deployment, implementation of our system and depicts the experiments results. Finally, Section VI concludes the paper and presents future works.

II. Related work

SDN, is a new paradigm that enables to implement new security concepts in comparison with traditional network. It uses many technologies in addition to its characteristics to insure an efficient protection of data and resources in the network communication. Blockchain technology can be integrated with SDN to efficiently secure the network. There are some works, which combine these two technologies to

provide an effective security in the network.

In [12], the authors propose an OpenFlow-based approach to secure blockchain nodes. This approach is implemented as a module on SDN controller. It uses the SDN components capabilities for filtering the incoming traffic attempt interacting with the blockchain nodes. It aims to secure private and consortium blockchain nodes against attackers and malicious attempt from one or multiples source IP address. The authors in [13] addressed the security of IoT networks. They propose a distributed secure SDN architecture for IoT network. In their scheme, the blockchain technique is used to secure the flow rules table and the security is automatically adapted to the threat landscape without the need of administrator.

In [14], the authors used backup controllers to detect compromised SDN devices. The backup controller is used to audit the online handling information of network update events collected from the primary controller and its switches. The collected information is analysed for detecting compromised devices. A blockchain-based distributed cloud architecture for IOT with SDN is considered in [15]. This approach provides a secure distributed fog node architecture that uses SDN and blockchain techniques to bring computing resources to the edge of the IoT network. The proposed architecture was designed to support high availability, real time data delivery, high scalability, security, resiliency, and low latency. Three layers are described the architecture of the proposed model, which are device, fog, and cloud layer. The work proposed in [16] studies the potential SDN vulnerabilities that might be used by attackers to launch controller and switch hijacking. This work presents mechanism for preventing misbehaving applications from hijacking controller. The work proposed in [17] presented another hijacking controller preventing scheme.In addition, another solution of a fault-tolerant SDN control plane is proposed in [18]. In this solution, network-related and application-related states are stored in a shared data store for smooth transition. In [19] and [20], the authors focused on the flow rules checking to detect the conflicting rules.

All the above related works didn't consider the man in the middle attack. Therefore, our solution comes to overcome this problem by avoiding false rules injection and securing the communication between the controller and the SDN data layer network elements.

III. Proposed Solution

In our proposed solution, we focus on the controller SDN security. We use blockchain technology to secure the communication between the SDN controller and the other network elements against False Flow Rule Injection (FFRI). Below, we present the components of our system and we describe its main concept.

A. System environment and its components

The system environment is considered as Software Defined Network mainly represented by its three main functional components (Fig.1). SDN Application contains various functionalities, such as network management, policy implementation, and other security services. SDN Controller represents logical centralized control software, it maintains global view of the network, and provides hardware abstractions to SDN applications. SDN Data layer elements component represent the physical layer and it contains the forwarding network devices used for forwarding traffic flows.

In this environment, SDN controller receives instructions from the SDN Application layer and relays them to the networking components. It may extract information about the network from the hardware devices and sends it back to the SDN Applications with an abstract view of the network.

The SDN Data layer elements forward and process the network data flow according to the flow rules sent from the SDN controller. Therefore, any false flow rule sent to the network devices has major impact on the forwarding and processing of the network data flow.

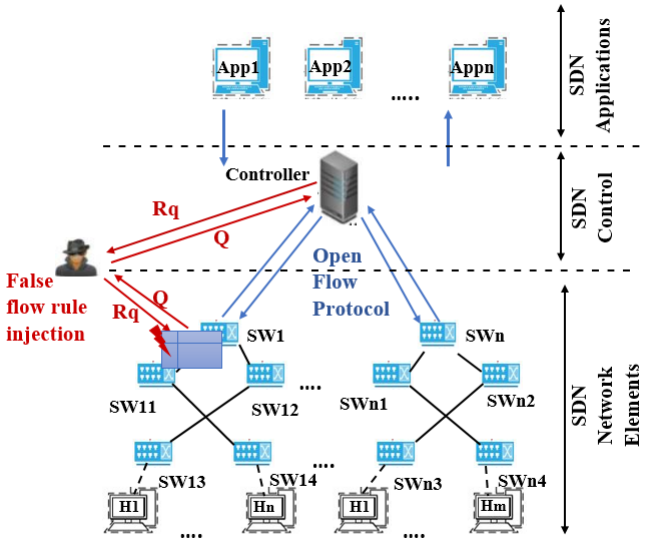

Fig. 1. SDN Architecture and threat model.

As shown in Fig.1, a malicious actor can realize man-in-the-middle attack by inserting himself as relay into a communication session between SDN controller and SDN network device. He could impersonate both parties and gains access to information that the two parties were trying to send to each other. This attack allows a malicious actor to secretly intercept, send and receive flow rules without being noticed.

B. Concept of the proposed solution

In SDN network, the security of the information exchanged between controller and the network devices of the forwarding plane is very important. To secure SDN network against false flow rules injection, we propose to protect the communication between SDN controller and forwarding devices using blockchain technology. Fig.2 shows the architecture of the proposed solution. In this architecture, a new trusted device is used. This device communicates with the SDN controller and the SDN network elements. Its principal role is the detection

of eventual FFRI. Our solution is divided into two parts, blockchain communication and FFRI detection approach.

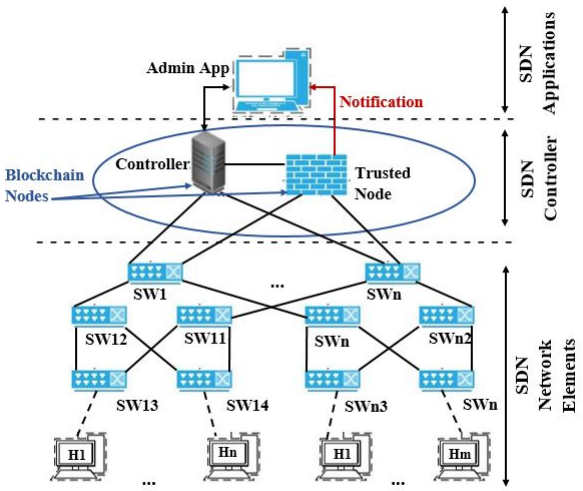

Fig. 2. Architecture of proposed System.

1) Blockchain communication: SDN controller and trusted device form the private blockchain network. This blockchain is opened only for these two nodes. When the controller sends flow rule to SDN network element via Openflow protocol [21], a copy of this flow rule is sent to the trusted node via blockchain. We use permissioned blockchain in which SDN controller can write new blocks into the chain and transact on the blockchain. The second node permissions are limited for reading the information on the blockchain. Moreover, this node can access to any SDN Data layer element and retrieve its flow rule table.

The block of blockchain contains the below information:

- Publishers: Identifier of SDN controller.
- Keys: The Publication identifier.
- Data: The flow rule of the transaction.
- Blocktime: The Time of the block creation.
- Txid: The Block ID.

The controller's data flows are composed of several recorded rules at the blockchain in a database composed as follow:

- ID: The identifier of the rule.
- TABLEID: The identifier of the flow table.
- DEVICEID: The identifier of the equipment on which the rule will be executed.
- TYPE: The rule type: input/ output.
- OUTPORT: The Output port number.
- INPORT: The Input port number.
- PRIORITY: The priority of the rule.
- MACSRC: The source mac address of flow.
- MACDST: The destination mac address of flow.

2) Detection and prevention approaches: To secure the communication between the controller and the network elements, we propose two approaches. The first approach detects and reports the network threats, while the second prevents and avoids the suspicious attacks. Fig.3 describes the main steps of the first approach.

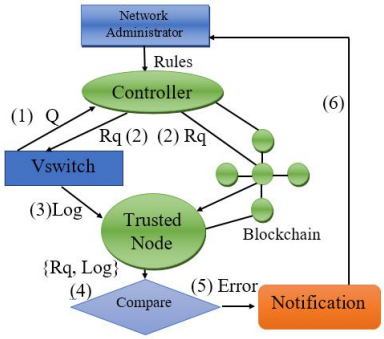

Fig. 3. Attack detection approach.

a) Attack detection approach: In this approach, as shown in the Detection Algorithm (Algorithm 1), when a new packet arrives to the VSwitch, if there is no rule corresponding to this packet in the flow table, the SDN Data layer element sends a request (Q) to the controller via Open Flow API, for selecting the adequate rules to this new flow. The controller receives the rules from network administrators, it processes the received request(Q), sends the request rule (Rq) to the VSwitch, and stores it in a trusted node, member of blockchain. This node hashes the traffic (request rule) into a block and distributes it to the other blockchain members. The request rule is a set of flow rules permitting the SDN Data layer elements to forward a given traffic according to these specific rules. Next, the SDN Data layer element sends the flow rules which it received from the controller to the trusted node (Firewall node (VMFw)). In the other hand, the VMFw node access to the blockchain and consults the flow rules sent by the controller and then it performs the needed comparison. For the first time, the flow rules should be matched in cases of similarity or unequal rules. In case of any dissimilarity, the VMFw node has to notify the network administrator. This approach could detect the eventual attacks but it cannot prevent them. The SDN Data layer elements execute the flow rule sent by the SDN controller without any previous check. To overcome this limit, we propose another approach to prevent the network attacks and malicious attempts.

b) Attack prevention approach: In this approach (Fig.4), the SDN controller sends the flow rule to the SDN Data layer elements and creates a copy of the same flow in the blockchain node as the previous approach. When SDN Data layer element receives the flow rule sent by the controller and before executing the rule, it has to wait the approval of the VMFw node. So, the SDN Data layer element transmits the received flow rule to the VMFw node. The VMFw access to the blockchain, compares the two rules and replays the SDN Data layer elements by giving him the execution agreement of the rule in case of similarity between the rules. Else, it notifies the administrator about the eventual attack. The Prevention Algorithm (Algorithm 2) illustrates more details of the proposed solution.

Algorithm 1 Detection Algorithm.

$Q, Rq = request$
$Q.sends()$; ▷ VSwitch send a request q to the controller.
$Q.processing()$; ▷ Controller processes q.
$Rq.transmiting1()$; ▷ Controller transmits rq to SDN Data layer elements and blockchain node.
$logs - vsw.transmiting2()$; ▷ VSwitch transmits its logs to a VM firewall VMFw.
$VMFw.recuperates(rq - archiv)$; ▷ VMFw recuperates rq from blockchain.
$VMFw.compares(rq - archiv, logs - vsw)$;
if ((rq-archiv.id !=logs-vsw.id) Or (rq-archiv.adrsrc != logs vsw.adrsrc) Or(rq-archiv.adrdst != logs-vsw.adrdst)) **then**
    $Threat.signaling()$; ▷ signaling a threat to administrator.
end if

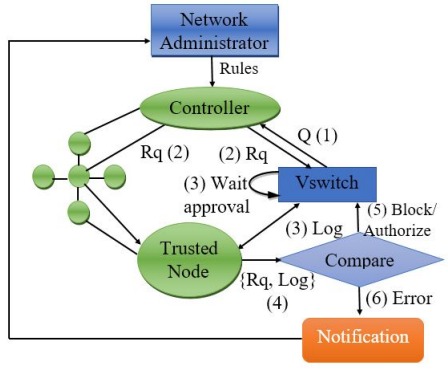

Fig. 4. Attack prevention approach.

IV. Deployment and Test Performance

A. Deployment environment

In order to test the performance of our solution, we deploy the environment presented in Fig.5. We use OpenStack [22] as a private Cloud. It provides a rapid deployment of cloud infrastructure with possibility of incorporating the SDN controller. We use Mininet emulator [23] to create a realistic virtual network. Mininet runs many hosts and Switches on a single OS kernel, its Switches support OpenFlow for highly flexible custom routing and SDN. We manage the SDN network by ONOS [24] controller. ONOS is Open Network Operating System which has been implemented for managing network operations following an SDN approach. We use Multichain platform [25] to construct the network blockchain, and we add a virtual machine which plays the role of the VMFw node. This machine checks the integrity of the flows. We create after the other components and we configure the needed service function chain that will be applied by the SDN controller. In this work, we evaluate the performance of the attack detection approach only. The second approach will be tested in upcoming work. Table 1 resumes network components of our environment.

Algorithm 2 Prevention Algorithm

$Rq, Q = request; Rp = received - packet;$
$Rq.sends()$;
$Rq.processing()$;
$Rq.transmiting1()$;
$vsw.block(Rp)$; ▷ VSwitch wait the approval and blocks the received packet from the controller.
$logs - vsw.transmiting2()$; ▷ vsw transmits its logs to a VM firewall VMFw.
$VMFw.retrieves(rq - BC)$; ▷ VMFw recuperates rq from blockchain.
$VMFw.compares(rq - BC, rq - vsw)$;
if ((rq-BC.id !=rq-vsw.id) Or (rq-BC.adrsrc != rq-vsw.adrsrc) Or(rq-BC.adrdst != rq-vsw.adrdst)) **then**
    $replay(Reject)$; ▷ reject the packet and signaling a threat to the administrator.
else
    $replay(rq.accept)$; ▷ request accepted.
end if
if (Vsw.respons = Rq.Reject) **then**
    $vsw.block(Rq)$;
else
    $vsw.authorize(Rq)$;
end if

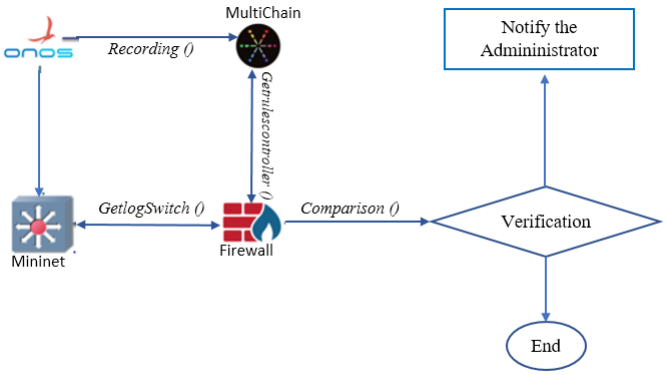

Fig. 5. The architecture components of the proposed system.

B. Implementation

In order to deploy our approach, we developed algorithms as shown in Fig.5 (Recording(), GetlogSwitch(), Getrulescontroller(), and comparison()). We use python language and Linux Ubuntu virtual machine.

- **The program *Recording()* :** Intercepts the flow rules sent from the ONOS controller to the VSwitch and records it on the blockchain. In order to access to the blockchain and create the needed blocks, the write authorization is assigned to this program by updating the configuration file multichain.conf. In this file, we can modify the parameters (blockchain name, port number, user name,

TABLE I
NETWORK COMPONENTS.

Network element	Description
Open stack	Ubuntu Server 64 bits
SDN Controller	ONOS, under Ubuntu virtual machine
Trusted node	Ubuntu virtual machine (Firewall)
Blockchain	ONOS, Firewall
MiniNet	network infrastructure consisting of Switches and hosts

password and IP address). Access to the blockchain is done by calling the remote RPC procedure using an open source library called Savoir using the following commands (Fig.6). Then, we record the controllers instructions by creating new block in the blockchain containing the rules sent from controller to VSwitch.

```
api = Savoir(rpcuser, rpcpasswd, rpchost, rpcport, chainname)
api.create('stream', 'C', True)
api.publish("C", Data, {'json':data})
```

Fig. 6. Blockchain access procedure.

- **the programs GetlogSwitch(), Getrulescontroller() and Comparison()**: are executed by virtual machine VMFw, to detect attacks and malicious attempts.
- textbfGetlogSwitch () accesses to Mininet via SSH connection and gets Vswitch logs. The retrieved data are inserted in a data dictionary for the comparison process.
- *Getrulescontroller()* recuperates the content of the last bloc of the blockchain and insert it in a data dictionary. This block contains the rules sent from controller to VSwitch.
- *The Comparison()* function compares the two data dictionaries (log SDN Data layer elements and controller rules). We compare the identifier, the identifier of table, the priority, the source and the destination MAC address of source and of destination, in addition to the type (Fig.7).

```
defFindItem(Liste, Chaine):
forelt in Liste:
ifelt.find(Chaine) == 0:
resultat = elt[1 + elt.index('='):]
defComparaison(Chaine_Type,Chain1,Chaine2):
Comparaison('IN_PORT',Flows['data'][i]['IN_PORT'],d['flows']
[j]['selector']['criteria'][0] ['port'])
if((len(list(set(ListeIDcontrolleur) - set(ListeIDSwitch))))
or (len(list(set(ListeIDSwitch) - set(ListeIDcontrolleur
))))): print "Attack,it misses " + str(len(list(set(
ListeIDcontrolleur) - set(ListeIDSwitch)))) + " Instructions
in Switchs"
```

Fig. 7. Implementation of the comparison function.

At the end of the Comparison, in case of dissimilarity a notification message will be sent to the network administrator.

C. Experiments and results

In this part, we present the results of our experimentation. We carried out different experiments to evaluate the accuracy, and efficiency of our solution.

1) System sizing: In this experiment, we evaluate the impact of the network size on the network performance. We increase the number of SDN Data layer elements from 10 to 100 and we measure the blockchain creation time, SDN Data layer elements logs recuperation time, blockchain log recuperation time, processing time and total execution time. Fig.8 shows the results of this experiment.

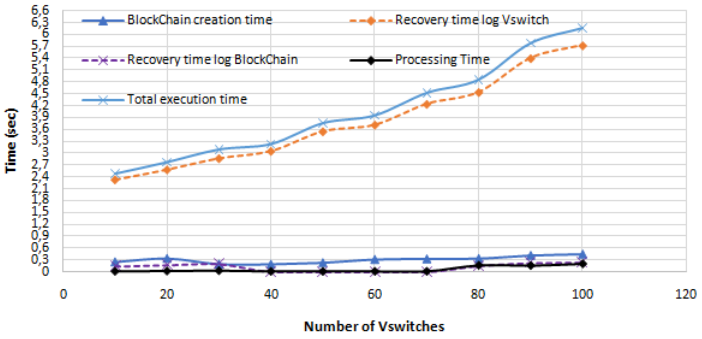

Fig. 8. Execution time according to Network size.

We note that the total execution time increases in parallel with the increasing of the number of VSwitches due to the SSH connection and SDN Data layer elements log recuperation from each VSwitch. The connection time between the SDN Data layer elements (VSwitch) and VMFw node could be enhanced according to the network connection type. In the other hand, we notice that the time required for creating the blockchain, the processing time, and the time of the recovery of flows through the blockchain are very low and are still stable. Then, we deduce that our program will not be able to manage very large infrastructure, due to the data recovery time of more than 30 seconds, in addition to the blockchain update time that causes errors in our program due to the recovery of outdated data. This problem could be resolved by using multi-controllers in SDN network.

2) Detection of attacks (False rules injection): In this experiment, we evaluate the ability of our program to detect number of false rules injected into our platform. For test reasons, we disconnect the controller in order to be able to send false rules to the VSwitch.

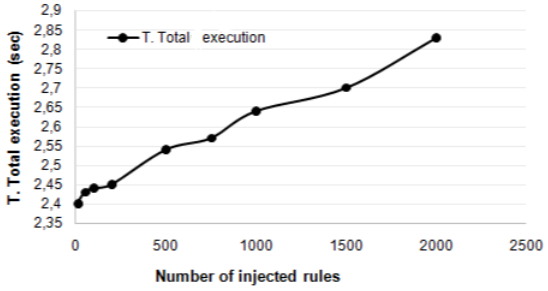

Fig. 9. Total execution time according to the number of injected instructions rules.

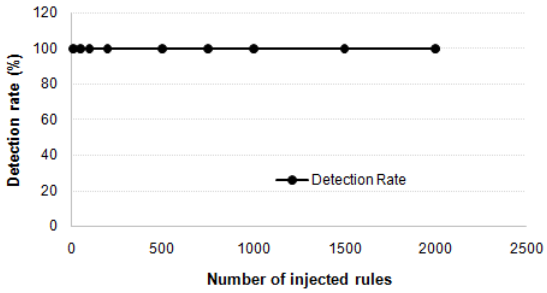

Fig. 10. False injected rules according to the detection rate.

The Fig.9 illustrates the total execution time according to the number of false injected rules. We observe that the total execution time increases in parallel with the increase of false injected rules. The VMFw node needs to check and compare each rule. This task takes a few processing time, and this time increases when the number of the checked rules increases. Fig.10 shows the impact of the number of false injected rules on the detection rate. We notice that the detection mechanism detects all the false injected rules. In this mechanism, the VMFw compares the flow rule sent by malicious node to the SDN Data layer elements with the same flow rule stored in the blockchain. An attack can be detected when the VMFw finds a difference between the blockchain flow rules and the SDN data layer elements flow rules. So, the malicious node cannot access to the blockchain and injects false flow rules. For this reason all the attacks have been detected.

V. Conclusion

We have proposed a new security solution for protecting SDN network using blockchain technology. The proposed solution integrates the quality of the blockchain to secure the SDN southbound communication. In this solution, we have proposed two approaches. The first approach detects false rules sent from the controller to the SDN Data layer elements after its execution by the VSwitch. The second approach verifies the integrity of the flow rules before its execution. In this paper, we have evaluated the first approach. The experiment results show that the proposed approach gives a good performance in term of attack detection and acceptable execution time. As extended future work, we plan to evaluate the second approach and enhance the execution time by proposing other extension features.

References

[1] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*. IEEE, 2012, pp. 877–880.

[2] D. W. Al Naimi, "Big data: A revolution that will transform how we live, work, and think," pp. 1143–1144, 2014.

[3] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International Journal of Communication Systems*, vol. 25, no. 9, pp. 1101–1102, 2012.

[4] F. Wortmann and K. Flüchter, "Internet of things," *Business & Information Systems Engineering*, vol. 57, no. 3, pp. 221–224, 2015.

[5] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[6] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[7] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Software-defined networking security: pros and cons," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 73–79, 2015.

[8] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

[9] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[10] M. Swan, *Blockchain Thinking: The Brain as a Decentralized Autonomous Corporation[Commentary]*. IEEE, 2015, vol. 34, no. 4.

[11] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, "Everything you wanted to know about the blockchain: Its promise, components, processes, and problems," *IEEE Consumer Electronics Magazine*, vol. 7, no. 4, pp. 6–14, 2018.

[12] M. Steichen, S. Hommes, and R. State, "Chainguarda firewall for blockchain applications using sdn with openflow," in *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*. IEEE, 2017, pp. 1–8.

[13] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "Distblocknet: a distributed blockchains-based secure sdn architecture for iot networks," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 78–85, 2017.

[14] H. Zhou, C. Wu, C. Yang, P. Wang, Q. Yang, Z. Lu, and Q. Cheng, "SDN-RDCD: A real-time and reliable method for detecting compromised SDN devices," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2048–2061, 2018.

[15] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for iot," *IEEE Access*, vol. 6, pp. 115–124, 2018.

[16] D. Kreutz, F. M. V. Ramos, and P. Veríssimo, "Towards secure and dependable software-defined networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013, The Chinese University of Hong Kong, Hong Kong, China, Friday, August 16, 2013*, 2013, pp. 55–60.

[17] C. Qi, J. Wu, H. Hu, G. Cheng, W. Liu, J. Ai, and C. Yang, "An intensive security architecture with multi-controller for SDN," in *IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2016, San Francisco, CA, USA, April 10-14, 2016*, 2016, pp. 401–402.

[18] F. A. Botelho, A. N. Bessani, F. M. V. Ramos, and P. Ferreira, "On the design of practical fault-tolerant SDN controllers," in *Third European Workshop on Software Defined Networks, EWSDN 2014, Budapest, Hungary, September 1-3, 2014*, 2014, pp. 73–78.

[19] E. Al-Shaer and S. Al-Haj, "Flowchecker: configuration analysis and verification of federated openflow infrastructures," in *3rd ACM Workshop on Assurable and Usable Security Configuration, SafeConfig 2010, Chicago, IL, USA, October 4, 2010*, 2010, pp. 37–44.

[20] S. Son, S. Shin, V. Yegneswaran, P. A. Porras, and G. Gu, "Model checking invariant security properties in openflow," in *Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, June 9-13, 2013*, 2013, pp. 1974–1979.

[21] R. Sherwood, M. Chan, G. A. Covington, G. Gibb, M. Flajslik, N. Handigol, T. Huang, P. Kazemian, M. Kobayashi, J. Naous, S. Seetharaman, D. Underhill, T. Yabe, K. Yap, Y. Yiakoumis, H. Zeng, G. Appenzeller, R. Johari, N. McKeown, and G. M. Parulkar, "Carving research slices out of your production networks with openflow," *Computer Communication Review*, vol. 40, no. 1, pp. 129–130, 2010.

[22] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[23] R. L. S. De Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, 2014, pp. 1–6.

[24] "Onos," http ://onosproject.org, accessed: 2019-01-03.

[25] "Multichain," http ://www.multichain.com, accessed: 2019-01-03.