BCSDN-IoT: Towards an IoT security architecture based on SDN and Blockchain

Original Scientific Paper

Younes ABBASSI
Hassan 2 University,
Faculty of Sciences Ben Msik, Computer Sciences Department
Casablanca, Morocco
younes.abbassi@univh2c.ma

Habib Benlahmer
Hassan 2 University,
Faculty of Sciences Ben Msik, Computer Sciences Department
Casablanca, Morocco
h.benlahmer@gmail.com

Abstract *– The Internet of Things (IoT) aims to create a digital world where any information system can expose, discover, understand and consume data and services for analysis, diagnosis, decision support and task automation in various domains such as healthcare, transportation, energy, industry, agriculture, etc. Faced with this diversity of applications and rapid evolution, infrastructures must be able to achieve high levels of security and confidentiality while being open, sustainable, and agile to adapt to the multiple requirements of applications.*

To meet these needs, new paradigms are emerging. These include the Software Defined Networks (SDN) paradigm, which offers the ability to dynamically program different applications and devices to provide end-to-end service chains. In parallel, the Blockchain paradigm is increasingly used in the Internet of Things, making distributed transactions between connected objects such as financial transactions or "smart contracts" possible.

Although the combination of these two paradigms (Blockchain/SDN) is a major issue for the success of the Internet of Things, paving the way for new business models and management/control of communication networks, there is not yet a specified/formalized architecture allowing the use of the "Blockchain" in SDN. In this research, a new architecture for a system combining blockchain and SDN for IoT security is proposed.

Keywords*: Internet of Things (IoT), Software-Defined Networking (SDN), OpenFlow, Blockchain, BCSDN-IoT Architecture*

1. INTRODUCTION

In 2030, it is announced that there will be more than 500 billion devices connected to the Internet with a variety of uses leading to security problems and an increase in traffic on the networks that will be estimated in Zeta (1021) bytes [1].

However, currently, the security architectures deployed in networks are mainly based on experience and work on wired networks. These architectures are mainly based on centralized equipment, whose main role is to control the information that is exchanged between the company's network and the outside world. It is therefore not possible to control the information exchanged between a terminal equipment that a user will connect to his computer. On a corporate network, users can connect their phone to their computer, via Bluetooth for example, and thus the computer becomes a new entry point to the network. With the Internet of Things (IoT), we have sensors, thermostats, webcams, watches connected to our phones, themselves eventually connected to the Internet or to our computers [1]. So how can we control the information coming from this large mass of heterogeneous devices?

With the increase in the number of these heterogeneous devices, the complexity in their administration is growing. This requires a verification of the coherence of the configurations of all the network devices of a company, for example the security rules and the user rights [2].

With the support of a great combination of modern technologies such as IoT, SDN, and Blockchain, as the number of connected things to the internet grows these days, managing and controlling IoT has become a very difficult task. SDN steps in to provide the IoT

network's adaptability and programmability without requiring existing implementations to change their design. It may also assess how the network affects the overall performance and efficiency of the network system, which is very useful when dealing with real-time transactions [4]. SDN is utilized in IoT applications to reduce response time and security concerns. In SDN, many controllers have recently been used instead of a centralized controller. The fundamental purpose of using multiple controllers is to balance the load between devices and controllers while minimizing packet loss. When the user of the SDN-IoT network need resources, they will be available immediately. In addition, utilizing an SDN controller, a network can be configured dynamically. One of the most common protocols used by SDN is OpenFlow [4].

Some other advanced technology is blockchain, a decentralized, emergent technology that can be combined with SDN-based IoT applications. The hash value is used to link various blocks together, and each block of the transaction is saved forever [5]. Combining this Blockchain technology will boost security and privacy. Several academics have proposed numerous clarifications to increase the network's performance, but they are unable to entirely cure the problem.

Although IoT, SDN, Blockchain technologies are combined to provide a better solution for any smart technology such as intelligent building, smart homes, smart cities, and smart grids [5]. These technologies can also provide reliable data transmission as well as communication in the networks [3].

However, the potential use of this disruptive technology spawn to each and every application that need to evolve from a centralized authorization entity acting as a trusted intermediary or sometimes a third-party verifiable trust anchor, towards a purely distributed authentication model.

Our goal in this paper is to give the reader particularly interested in IoT security, a proposal for a new security architecture combining SDN and Blockchain technologies, with the aim of improving, and simplifying the deployment of IoT security.

The remainder of the paper is laid out as follows: In section 2, we go over some background information before introducing IoT, SDN, and Blockchain, as well as their designs. Then, in section 3, we describe our proposed BCSDN-IoT architecture, its operation, and analysis and alert generation. In section 4, we conduct an implementation of the BCSDN-IoT solution in virtual through the open source solution OpenDayLight, starting with its installation, the realization of the BCSDN-IoT architecture and the simulation of some attacks. And finally in section 5 we conclude our article with some perspectives.

2. BACKGROUND & RELATED WORKS

2.1 INTERNET OF THINGS

The Internet of Things (IoT), as well as the Internet of Everythings (IoE) in a larger sense, is a relatively new concept. It is considered a major technological and economic innovation in the industry of new information technologies and communication.

The IoT does not have a unique definition but generally speaking, It is characterized as a broadening of the current Internet to include all objects that can communicate directly or indirectly with electronic equipment that is also linked to the Internet.

The International Telecommunication Union [7] defines the Internet of Things as: "A global infrastructure for the information society, which enables advanced services by interconnecting objects (physical or virtual) through existing or evolving interoperable information and communication technologies".

IoT devices are typically sensor nodes, RFID (Radio Frequency IDentification) tags and wireless communication devices connected to the Internet in a smart environment [2]. These devices are very diverse (phone, watch, refrigerator...) and are now widely used in everyday life.

With the exponential development of these connected objects with heterogeneous characteristics, the networks of the future must evolve towards new architectures to adapt to the increase in traffic and ensure their security. Security is one of the issues of today's Internet, as there are more and more intelligent security attacks to deal with. In addition, security attacks for IoT are more difficult to handle due to the minimal energy storage, data and processing capacity that are not suitable for existing network security mechanisms based on firewall and IDS/IPS [21]. The concept of IoT is relatively simple but there are many problems because these connected devices do not have enough capacity to handle the communications and processing associated with the applications.

Architecture: The most commonly used IoT architecture for SDN solutions, and as shown in Figure 1, is made up of three layers: the perception layer, the network layer and the application layer [8].

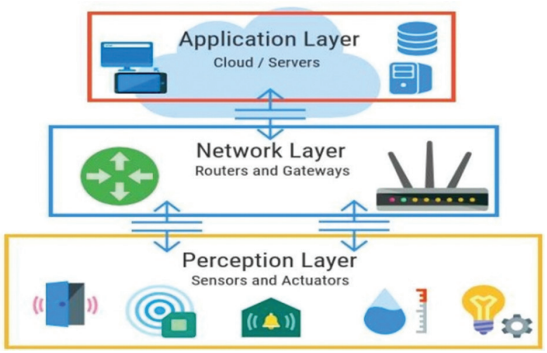

Fig. 1. The architecture of IoT

Perception layer: it is at this level that the collection of information takes place. Various devices and devices help it in this, such as smart cards, readers and sensors, RFID tags, etc.

It is personalized by a function that allows it to detect the whole object in order to acquire information about it at any time and place, through the RFID system. (EPC) Electronic Product Code is a unique identifier that distinguishes each object in the IoT infrastructure, it generates a sequence of numbers giving an idea about the producer of the object, its production date and the expiry date.... [8].

Network layer: This layer allows the sending of the required information from the previous layer to the internet through machines, wired or wireless network equipment. As a transport layer, digital data is transported reliably [8].

Application layer: or also known as the process layer analyzes the information received and makes control decisions to perform its intelligent processing function by connecting, identifying and controlling objects and devices. Intelligence assets use intelligent computing technologies such as cloud computing and process information for intelligent control, as well as the tasks that must be completed and when they must be completed [8].

2.2 SOFTWARE-DEFINED NETWORKING

SDN (software-defined networking) is a new network architecture paradigm that describes a control plane that is completely separate from the data plane. According to the ONF (Open Network Foundation) [9] SDN is an architecture that separates the control plane from the data plane, and centralizes all network intelligence [27] in a programmable entity called "Controller", in order to manage several elements of the data plane (e.g. switches or routers, etc.) via APIs (Application Programming Interface).

More concretely, we can say that a network architecture follows the SDN paradigm if, and only if, it verifies the following points:

- The control plane is completely decoupled from the data plane; this separation is materialized through the definition of a programming interface (Southbound API)
- All network intelligence is externalized in a logically centralized point called the SDN controller, which offers a global view on the entire physical infrastructure.
- The SDN controller is a programmable component that exposes an API (NorthboundAPI) to specify control applications.

Architecture: A traditional network is generally composed of interconnection equipment such as switches and routers. This equipment incorporates both the transmission and control parts of the network. In this architecture model, it is difficult to develop new services because of the strong coupling between the control plane and the transmission plane.

In order to open the network equipment to innovations, the SDN architecture was born. It allows decoupling the control part from the transmission part of the interconnection equipment [29]. As depicted in Figure 2, the SDN is made up of three layers and communication interfaces.

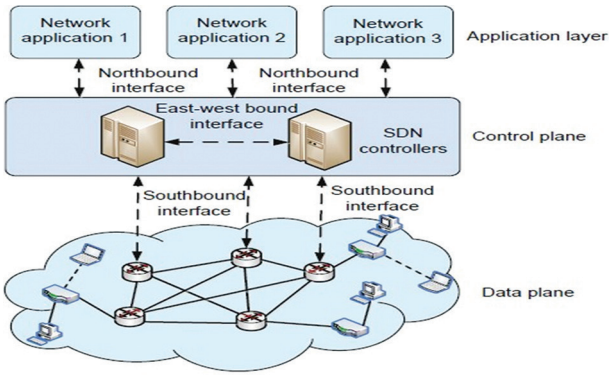

Fig. 2. The architecture of SDN

We describe in the following these layers, as well as the communication interfaces:

- **The transmission layer:** also called "data plane", it is composed of routing equipment such as switches or routers, its main role is to transmit data and collect statistics.
- **The control layer:** also called "control plane", it is mainly composed of one or more SDN controllers, its role is to control and manage the infrastructure equipment through an interface called 'south-bound API'.
- **The application layer:** represents the applications that enable the deployment of new network functionalities, such as traffic engineering, QoS, security, etc. These applications are built through a programming interface called 'northbound API'.

Communications Interfaces

There are three main types of interfaces, which allow controllers to communicate with their environment: South, North and East/West interfaces

Southbound APIs: are the interfaces that allow the SDN controller to communicate with infrastructure layer devices like switches and routers.

The most widely used protocol, and the most deployed as a Southbound interface is the OpenFlow protocol, which has been standardized by the ONF, its latest version is 1.5 [10], more details on this protocol will be given in the next section. There are now other southern interface alternatives, such as ForCes [11], or

Open vSwitch Database (OVSDB) [12], but the OpenFlow protocol is currently the de facto standard, which is widely accepted and spread in SDN networks.

Nord interfaces: are used to program transmission devices, exploiting the network abstraction provided by the control plane. It is noted that unlike the Southbound API which has been standardized, the North interface still remains an open question.

While the need for such a standardized interface is a considerable debate within the industry, the advantage of an open Northbound API is also important, as an open Northbound API allows for more innovation and experimentation. Several implementations of this interface exist, each of which offers very different functionality. The RESTful [13] is considered the most widespread North API in SDN.

East/West interfaces: are communication interfaces that allow communication between controllers in a multi-controller architecture to synchronize the network state. These architectures are very new and no inter-controller communication standard is currently available.

2.3. BLOCKCHAIN

Blockchain technology emerged in early 2009 with the crypto-currency Bitcoin (BTC). Bitcoin users use a variable public key (PK) [14] to generate transaction information and broadcast it to the network for transferring funds. Transaction information is stored by all users in its own block. Once the block is full, a network mining process is performed; the hash value of the block is calculated, and the encrypted information and blocks are added to the blockchain [15]. To mine the cryptographic hash value of a block, certain nodes in the network, known as miners, compete to solve a proof of work called the cryptographic resource consumption puzzle (POW) [28]. The node that solves the puzzle first and gets everyone's approval is considered to have mined the block. This is because blockchain technology maintains all transaction data counts among all members, and all members update the counts simultaneously to maintain completeness when new transactions occur [16] [23]. The Internet and encryption technologies are the underlying technologies that allow all members to verify the reliability of each transaction to resolve a single point of failure caused by a traditional third-party authorized transaction. Because the blockchain is a peer-to-peer (P2P) network [17], the transaction is free of unauthorized third-party charges. As everyone keeps their transaction information up to date, the hacking effect of single point records is very limited, and it frequently fails. In addition, users of a blockchain system can openly access transaction records and reduce the costs of monitoring transactions. Since the hash value stored in each block peer is affected by the block peer is affected by the value of the previous block, forgery and modification of data requires modification of the entire chain [18] and the amount of computation at one point is far behind the computation of the entire network. As a result, forgery is almost impossible.

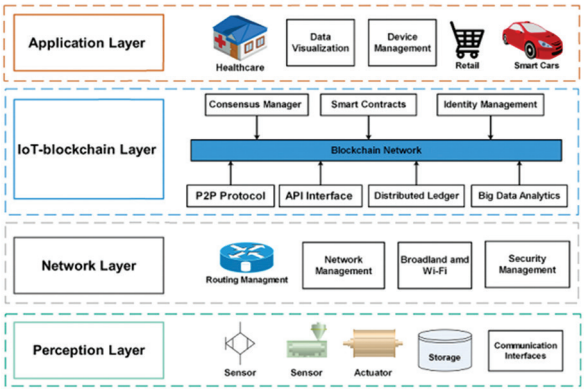

Fig. 3. The architecture of IoT with Blockchain

2.4 RELATED WORKS

Several researchers have recently addressed emerging leading technology such as IoT, SDN, Blockchain, and other smart technologies in today's world [19, 20, 21, and 22]. In this section, some literature reviews of recent works have been mentioned which are given below:

Table 1. State of the art.

Research Work	Used			Summary contributions and features
	IoT	SDN	Block chain	
M. J. Islam, M. Mahin, S. Roy, B. C. Debnath, and A. Khatun.[19]	✓	✓	*	Presented a distributed black net with SDN-IoT architecture for smart cities and addressed the cluster head selection scenario
M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke [20]	✓	*	✓	Provided several overviews of the Blockchains application domain in IoT, e.g: Vehicle Internet, Energy Internet, Cloud Internet
P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, [21]	✓	✓	✓	Proposed a literature combination between Blockchain and SDN for IoT networks and presented flow rule table for validation of blocks as well
C. Qiu, F. R. Yu, F. Xu, H. Yao, and C. Zhao [22]	✓	✓	✓	Proposed an imminent permitted blockchain-based consensus in distributed SDIoT and also efficiently used a novel dueling deep Q-learning approach.

3. BCSDN- IoT PROPOSED ARCHITECTURE COMBINING SDN AND BLOCKCHAIN FOR IoT SECURITY

Based on the analysis in the previous section for the rapidly growing IoT networks created by new communication paradigms, we observed that the current distributed network architecture, protocols, and techniques are not designed to meet the design principles required for future challenges and satisfy new service requirements. Today, organizations need a unique distributed security architecture that includes powerful network security devices that provide real-time proactive protection and high performance to address the design principles analyzed. In this section, we provide the distributed secure SDN architecture,BCDSN-IoT architecture, its workflow, and a mechanism for updating high-performance availability flow rule tables play an important role in a distributed blockchain network.

3.1. BCSDN-IoT APPROACH

BCSDN-IoT adopts distributed secure network control in the IoT network using the concept of blockchain technology to improve security, scalability, and flexibility without the need for a central controller. Figure 5 shows the overall view of the proposed architecture. In the proposed architecture, all controllers in the IoT network are interconnected in a distributed blockchain network fashion so that each IoT transmitting device in the network can communicate easily and efficiently. Each local network view includes an IDS/IPS (Intrusion Detection System/Intrusion Prevention System) service. By putting an IDS module on each controller, the BCSDN-IoT architecture not only enables operational flexibility, but also proactive and reactive incident prevention based on the repeating threat environment, which is fast evolving, dynamic and high performing. It provides an agile, modular and secure network infrastructure. Protections must dynamically adapt to the threat landscape without requiring security administrators to manually process large numbers of notifications and approvals. These assurances must be well-coordinated across the broader IoT environment, and the architecture must adopt a protection posture that uses both internal and external sources constructively.

Our solution is inspired by the security grid concept and our intelligent firewall approach to improve security in a conventional network and extend it to the IoT.

In this approach, we propose a collaborative security solution with a distributed controller architecture coupled with IDS. We have opted for a distributed SDN architecture distributed SDN architecture because a centralized architecture with a single controller increases the danger of network in the event of a denial of service (DoS) attack, there will be a service outage. For example, if the threat is only on one machine, it is not critical and isolating the machine can be a solution, but if the single controller is compromised, the whole network is at risk. The use of multiple controllers therefore creates redundancy, ensures high availability and reduces network latency.

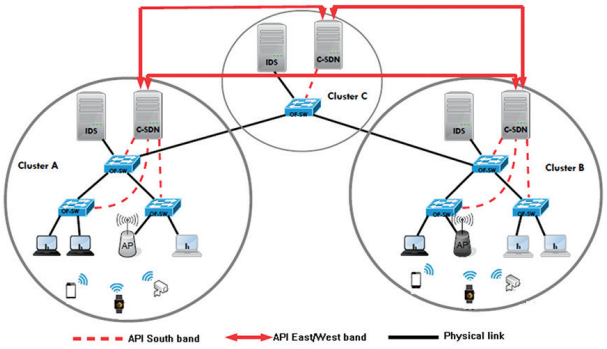

Fig. 4. Distributed Routing Cluster for SDN

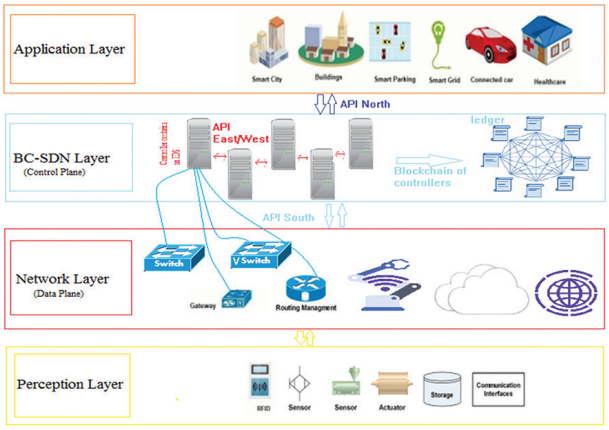

Fig. 5. The BCSDN-IoT architecture proposed

As shown in Figure 4, our solution consists of one or more clusters. Each cluster is composed of one or more network devices that are responsible for the interconnection of devices including connected objects. Within each cluster, an SDN controller manages the OpenFlow network. Each SDN controller is coupled to an IDS. The IDS is responsible for detecting intrusions into the network perimeter of each cluster. In other words, a cluster is an SDN domain in which we use an OpenFlow network with an SDN controller and an IDS to manage the security domain, which we call the zone of trust. To form a trust zone, all the equipment and devices in this zone must be fully secured. This security work is done by the controller and IDS pair. The SDN controller acts as an intelligent firewall for the trust zone and has security rules specific to the security needs of the cluster. These rules are programmed by an administrator. They can be distributed to other trust zones if the security need is the same through the East-West API.

Our approach allows not only to manage security in a totally decentralized way through a local management of security by the SDN/IDS controller couple, but also that the controllers exchange information on the threats detected in their respective clusters.

The SDN controller is the central element for security management in each trust zone. It has a global view of the network, manages traffic and distributes security policies to the network devices in its own cluster.

Before the SDN controller can isolate the threat in each cluster, it must be detected. That is why we use an IDS to solve this problem. In practice, this can be done by setting up an IDS like snort or other.

3.2 ANALYSIS, DETECTION AND ALERT GENERATION

To achieve this, we used an IDS to listen to all network traffic, analyze and detect malicious flows.

The IDS analyzes the network data and detects anomalies or attack patterns predefined by the blockchain network administrators. This detection is mainly based on the analysis of the network and transport layer packet headers but also on the packet content. To detect a malicious flow, the IDS mainly uses two analysis methods, namely the signature-based detection method which allows to detect known patterns in the analyzed data, or the behavioral detection method which detects deviations of a behavior from a normal profile. In both cases, the IDS compares the analyzed data to a reference described either by a signature or by a normal profile. Once the data is analyzed, the IDS can generate an alert in the form of a log file in case of malicious flows.

4. IMPLEMENTATION OF THE BCSDN-IoT SOLUTION

Our implementation model is entirely realized in a virtual environment with open source tools.

4.1 INSTALLING OPENDAYLIGHT

The OpenDaylight controller is an open source network operating system software developed in Java and managed by the Linux Foundation. It is based on a modular architecture and exchanges with SDN applications using the Northbound API. OpenDaylight communicates with network devices using its Southbound API. The most commonly used Southbound API in SDN is OpenFlow.

To experiment with our solution, we created a virtual machine with 2 CPUs and 16GB of RAM with an Ubuntu 16.04 operating system on the VMware platform. Then we installed on this machine an OpenDaylight SDN controller Beryllium- SR4 version.

Once OpenDaylight was installed, we added features such as odl-l2switch-switch, odl-dlux-all and odl-restconf to support Layer 2/3 switches, web interface and communicate with applications via the REST API. It is also important to enable OpenFlow version 1.3 by adding the -of13 option on the launch script file, as OpenFlow version 1.0 is implemented on the OpenDaylight controller by default. OpenDaylight provides several types of features to use as needed.

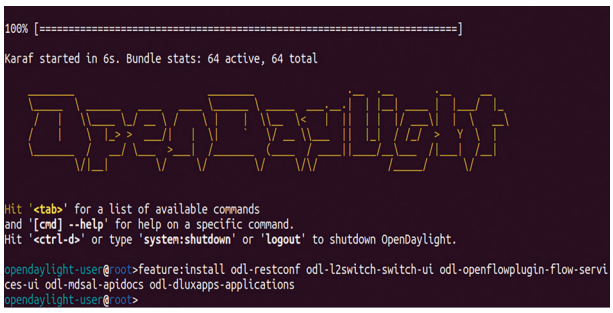

Fig. 6. Installing OpenDaylight

Once OpenDaylight was installed, we added features such as odl-l2switch-switch, odl-dlux-all and odl-restconf to support Layer 2/3 switches, web interface and communicate with applications via the REST API. It is also important to enable OpenFlow version 1.3 by adding the -of13 option on the launch script file, as OpenFlow version 1.0 is implemented on the OpenDaylight controller by default. OpenDaylight provides several types of features to use as needed.

To make a Layer 2/3 OSI routing decision, the OpenDaylight controller knows the network topology, as well as the devices that are connected with their identifiers (IP addresses and MAC addresses). Using OpenFlow 1.3, the OpenDaylight controller configures an OVS switch and manages and updates the OpenFlow network.

4.2 REALIZATION OF THE ARCHITECTURE

Most of the works in the literature use the mininet network simulator to experiment the SDN network. We have chosen to use virtual machines in a production environment on VMvare platform, to be in a real use case.

To realize our virtual network architecture, we created a second virtual machine with an Ubuntu 16.04 operating system, 2 virtual CPU and 16GB of RAM on a VMvare platform. On this machine, we installed an OpenFlow 1.3 compatible virtual switch (OVS version 2.6.0) and Qemu (Quick Emulator), an open source virtual machine emulator on x86 architecture.

The OVS is an open source software implementation of an Ethernet switch with the particularity of being multilayer and distributed. It is designed to work as an OSI level 2/3 switch in virtual machine environments supporting different protocols and standards, including the OpenFlow protocol. In our work, it has allowed us to make client virtual machines communicate with each other. Qemu is used to emulate our client machines with an Alpine Linux operating system, an ultra light distribution of Linux with 48MB of RAM. We used the basic qemu-img tool to create and manage disk images. The qcow2 format is used in this work because it integrates more features like compression and encryption.

Then, we wrote a bash script to launch several qemu client virtual machines with the possibility to manage them remotely. The same script allows to launch the OVS to interconnect the Alpine Linux virtual machines

and to create the link between the OpenFlow switch and the OpenDaylight controller, to allow the latter to control the network via the OpenFlow 1 protocol.

A dynamic allocation of IPv4 addresses in DHCP of the virtual machines clients of the network is made by the same code. This is how we set up our OpenFlow network with the possibility of scaling up just by varying the number of virtual machines and number of virtual machines and OVS desired.

4.3 SECURING THE LINK BETWEEN THE OPENDAYLIGHT CONTROLLER AND THE OVS

As discussed earlier, the communication channel between the OVS and the OpenDaylight controller is not encrypted by default, which means that encryption of OpenFlow exchange messages between these two elements of the SDN network does not run automatically. In addition, some controllers do not even support TLS for encrypting communications between the SDN switch and the controller. A hacker can exploit this lack of security on the OpenFlow channel to attack the network and conduct malicious actions. This is extremely dangerous if the hacker gains access to the controller that would give him control over the entire network. With a grip on the controller, the hacker can remove OpenFlow switches, modify OpenFlow rules in the switch, capture sensitive traffic and monitor how the controller handles OpenFlow packets. For this reason, SSL/TLS encryption of OpenFlow message exchanges on the channel between the OVS and OpenDaylight is required.

The encryption of OpenFlow messages between the OVS and OpenDaylight is done using an SSL/TLS connection, based on the Public Key Infrastructure (PKI) model.

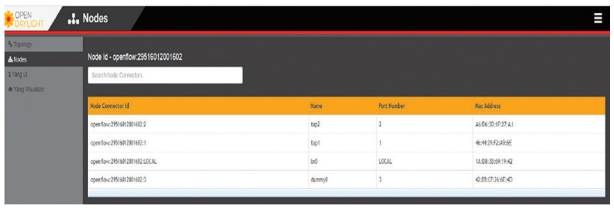

Fig. 7. Node Inventory on the OpenDaylight Beryllium-SR4

Using the OpenSSL encryption toolkit, we generated a keyStore, a file containing the controller's private and public keys. Then, the key file is imported into a JKS format key file, adapted to be configured on the OpenFlow configuration file of the OpenDaylight controller.

4. 4 SOME EXAMPLES OF SIMULATED ATTACKS

To simulate an attack and see if the IDS detects it or not, we installed the Nmap tool on one of the client virtual machines. Then, we successively launched a denial of service attack, a port scan and an IP address spoofing with the specific Nmap command on a case by case basis.

- **Denial of service**

The objective here is to detect and block attempts to saturate a target machine with DoS attacks using the ICMP protocol. We proceeded to send ICMP requests to the second machine of our network, in order to see if the Snort IDS reacted by detecting the unwanted flows. With this example, we found that after this ICMP request, our Snort IDS detected and saved a log file on the specific directory of the Snort server. This type of attack attempt can make the controller or a machine unavailable to its users. It interrupts or suspends network services temporarily or indefinitely.

With the proposed solution, it is possible to block the communication of malicious nodes in an automated way.

- **Port scan**

In this case, the goal is to detect port scan attempts on TCP and UDP protocols and to block these requests from the same source with the Nmap tool. Nmap is an open source port scanning software designed to detect open ports and, more generally, to obtain information about the operating system of a remote computer. To find out which ports are open on a machine, Nmap sends a packet to all ports on the target machine and analyzes the responses.

To simulate port scanning, we installed the Nmap tool on one of the Linux host alpine machines on our network. Then, we launched a port scan on one of the machines of the network with the specific command (nmap -p "*" Ip address target machine) and in the same way, Snort detected this attack attempt and recorded the corresponding log.

- **IP or MAC address spoofing**

MAC address spoofing is when a malicious attacker attempts to spoof a legitimate MAC or IP address in order to send packets to the network, using a trusted address. MAC/IP address replication forces systems to believe that the source is trustworthy.

In the same way as port scanning, we experimented with Nmap, and through the specific command (nmap spoof-mac target machine MAC address or target machine IP address), IP and MAC address spoofing and found that Snort detected the threat and logged the associated log.

We noticed that Snort detected all the attacks and saved the corresponding files in the log directory. This procedure can be extended to other types of more complex and intelligent threats.

5. CONCLUSION

In this paper, based on an analysis of the challenges faced by large-scale IoT networks due to new communication paradigms, BCSDN-IoT, a novel distributed secure IoT network architecture composed of an SDN backbone using blockchain technology, has been proposed to address current and future challenges and

satisfy new service requirements. BCSDN-IoT improves the performance and capacity of a system. The primary role of the BCSDN-IoT model is to generate and deploy protections, including threat prevention, data protection, and access control, and mitigate network attacks such as cache poisoning/ARP spoofing, DDoS/DoS attacks, and detect security threats. The BCSDN-IoT approach also focuses on minimizing attack window time by allowing IoT forwarding devices to check and download the most recent flow rule table if necessary. The performance evaluation is based on the scalability, defense effects, accuracy rates and performance overhead of the proposed model.

6. REFERENCES

[1] H. Hamed, D. Ali, M. Reza, A. Mohammed, K. Hadis, "A Survey on Internet of Things Security: Requirements, Challenges, and Solutions", Internet of things, Vol. 14, 2021, 100129.

[2] H.W. Rolf, "Internet of Things – Need for a New Legal Environment?", Computer Law & Security Review, Vol. 25, 2009, pp. 522–527.

[3] P. Sanghera, T. Frank, "How to Cheat at Deploying and Securing RFID", Chapter 15 - RFID Security: Attacking the Backend", How to Cheat, 2007, pp. 311-321.

[4] J. Sweta, C. Pruthviraj, S. Ayushi, "The Fundamentals of Internet of Things: Architectures, Enabling Technologies, and Applications", Healthcare Paradigms in the Internet of Things Ecosystem, 2021, pp. 1–20.

[5] O. Flauzac, C. Gonzalez, F. Nolot, "New Security Architecture for IoT Network", Procedia Computer Science, Vol. 52, 2015, pp. 1028–1033.

[6] A. I. Sanka et al. "A Survey of Breakthrough in Blockchain Technology: Adoptions, Applications, Challenges and Future Research", Computer Communications, Vol. 169, 2021, pp. 179–201.

[7] Internet of Things Global Standards Initiative, ITU 2021.

[8] A. Mayuri, T. Sudhir, "Internet of Things: Architecture, Security Issues and Countermeasures", International Journal of Computer Applications, Vol. 125, No. 14, 2015, pp. 1–4.

[9] Software-Defined Networking (SDN) Definition Open Networking Foundation, https://www.opennetworking.org/sdn-definition (accessed: 2018)

[10] Openflow-switch-v1.5.1, https://www.opennetworking.org/wpcontent/uploads/2014/10/openflow-switch-v1.5.1 (accessed: 2018).

[11] J. Halpern, S. J. Hadi, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", Internet Engineering Task Force , 2010.

[12] B. Pfaff, B. Davie, "The Open vSwitch Database Management Protocol", RFC Editor, RFC7047, 2013.

[13] F. R. Thomas, "Architectural Styles and the Design of Network-based Software Architectures", Information and Computer Science, 2000, pp. 180.

[14] C. Konstantinos, D. Michel, "Blockchains and smart contracts for the internet of things", IEEE Access, Vol. 4, 2016, pp. 2292–2303.

[15] B. Mandrita, L. Junghee, R. Kim-Kwang, "A Blockchain Future for Internet of Things Security: A Position Paper", Digital Communications and Networks, Vol. 4, No. 3, 2018, pp. 149–160.

[16] G. Sangeeta, A. Kavita, "Essentials of Blockchain Technology for Modern World Applications", Materials Today: Proceedings, 2021.

[17] T. Christos, B. Konstantinos, G. Panagiotis, A. Stavros, D. Tasos, "Malicious threats and novel security extensions in p2psip", Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops, Lugano, Switzerland, 19-23 March 2012, pp. 746–751.

[18] S. Jie, Z. Pengyi, A. Mohammed, B. Yubin, Ge YU, "Research Advances on Blockchain-as-a-Service: Architectures, Applications and Challenges", Digital Communications and Networks, 2021.

[19] M. J Islam, M. Mahin, S. Roy, B. Debnath, A. Khatun, "DistBlackNet: A distributed secure black SDN-IoT architecture with NFV implementation for smart cities", Proceedings of the International Conference on Electrical, Computer and Communication Engineering, Cox'sBazar, Bangladesh, 7-9 February 2019, pp. 1–6.

[20] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, H. Janicke, "Blockchain technologies for the Internet of Things: Research issues and challenges", IEEE Internet Things, Vol. 6, No. 2, 2019, pp. 2188–2204.

[21] P. Sharma, S. Singh, Y. Jeong, J.H. Park, "DistBlockNet: A distributed blockchains-based secure SDN architecture for IoT networks", IEEE Communications Magazine, Vol. 55, No. 9, 2017, pp. 78–85.

[22] C. Qiu, F. R. Yu, F. Xu, H. Yao, C. Zhao, "Permissioned blockchain based distributed software-defined industrial Internet of Things", Proceedings of the IEEE Globecom Workshops, 2018, Abu Dhabi, United Arab Emirates, 9-13 December 2018, pp. 1-7.

[23] R. Kumar, R. Sharma, "Leveraging Blockchain for Ensuring Trust in IoT: A Survey", Journal of King Saud University - Computer and Information Sciences, 2021.

[24] S. A. Latif et al, "AI-Empowered, Blockchain and SDN Integrated Security Architecture for IoT Network of Cyber Physical Systems", Computer Communications, Vol. 181, 2022, pp. 274–283.

[25] S. Rathore, B. W. Kwon, J. H. Park, "BlockSecIoTNet: Blockchain-Based Decentralized Security Architecture for IoT Network", Journal of Network and Computer Applications, Vol. 143, 2019, pp. 167–177.

[26] P. Podili, K. Kataoka, "TRAQR: Trust Aware End-to-End QoS Routing in Multi-Domain SDN Using Blockchain", Journal of Network and Computer Applications, Vol. 182, 2021, 103055.

[27] P. P. Ray, N.j Kumar, "SDN/NFV Architectures for Edge-Cloud Oriented IoT: A Systematic Review", Computer Communications, Vol. 169, 2021, pp. 129–153.

[28] Md. A. Uddin et al. "Blockchain Leveraged Decentralized IoT EHealth Framework," Internet of Things, Vol. 9, 2020, 100159.

[29] V. Balasubramanian, M. Aloqaily, M. Reisslein, "An SDN Architecture for Time Sensitive Industrial IoT", Computer Networks, Vol. 186, 2021, 107739.