Future Internet of Things with Software Defined Networking

V.Jeyakumar
Asst., Professor
Department of Computer Science
KG College of Arts and Science
Coimbatore, Tamil Nadu, India

ABSTRACT

Future Internet of Things (IoT) will hook up with the web billions of heterogeneous smart devices with the capacity of interacting with the environment. Therefore, the proposed solutions from an IoT networking perspective must take under consideration the scalability of IoT nodes also because the operational cost of deploying the networking infrastructure. This will generate an enormous volume of knowledge, which poses an incredible challenge both from the transport, and processing of data point of view. Moreover, security issues appear, thanks to the very fact that untrusted IoT devices are interconnected towards the aggregation networks.

In this paper, we propose the usage of a Software-Defined Networking (SDN) framework for introducing security in IoT gateways. An experimental validation of the framework is proposed, leading to the enforcement of network security at the network edge.

Keywords—*IoT, SDN, Security, Analytics*.

I.INTRODUCTION

Billions of objects are going to be connected to the web within the coming years. Therefore, it's expected a true revolution on the quantity of knowledge gathered and shared. This is referred to as the web of Things (IoT). Everyday objects, like home appliances, lampposts, traffic lights or irrigation outlets, are some samples of smart things. They are equipped with several sensors generating data, which then should be gathered and analyzed.

Cloud computing refers to the power to store and access data and programs over the web. It is a service offered by centralized large scale data centers, which could be geographically distributed. Instead, fog computing may be a new paradigm for a decentralized and distributed computing infrastructure during which application services are handled at the network edge. Its goal is to enhance efficiency and reduce the quantity of knowledge that must be transported to the cloud for processing, analysis and storage [1].

The integration of IoT with fog and cloud computing may be a valuable solution thanks to the functionality of computing, storage and networking resources at the sting of the network, thus allowing fast interaction with the info and low latency. Fog and cloud computing are expected to permit the info storage and processing from billions of smart things and IoT gateways. IoT gateways are key enablers for IoT and typically consists of small gateways which are ready to interconnect distributed wireless sensors, interconnected through wireless sensor networks (WSN), and acting as an online gateway for the interconnected devices.

Software Defined Networking (SDN) [2] is predicted to be a key enabler for subsequent generation networks, the so-called 5G (5th generation of wireless systems), which can got to integrate both IoT services alongside traditional human- based services. during this context, SDN enables a worldwide orchestration of distributed cloud, heterogeneous network and IoT resources required so as to: a) Transport the large amount of knowledge generated at the terminals, sensors, machines, nodes, etc., to any distributed computing node, edge, or core data center; b) Allocate computing, storage and network resources, and; c) Process the collected data (Big Data) and make the right decisions (cognition).

One of the most important challenges which presents IoT to network administrators, is that the ability to gather data and conduct analysis to supply a positive user experience on the go. SDN is in a position to redirect

traffic automatically when needed, which significantly improves the IoT applications. Through orchestration virtual network, storage and computing resources are provisioned and instantly delivered for the analysis of knowledge.

However, thanks to the very fact that untrusted IoT devices might be interconnected towards the aggregation networks or external malware could be applied to the network [3] security issues may arise. it's during this context, SDN offers a plethora of solutions to reinforce security.

The technical solution proposed during this paper provides an agile answer on the way to overcome these problems and improve the IoT security within software-defined networks. The proposed work consists within the design and validation of an SDN-enabled security framework in an experimental platform (network environment). Moreover, we've designed, evaluated experimentally validate an intrusion detection algorithm as an SDN application.

This paper is organized as follows.
In Section II, an introduction on the State of the Art on IoT, SDN and Security is provided. Section III proposes an SDN-enabled security architecture, which leverages the available real-time network flows information so as to require decisions regarding the provided security at the sting . An intrusion detection algorithm is proposed to validate the architecture design and it's evaluated in Section IV, during a simulated environment. In Section V, the experimental evaluation and results are provided for the proposed security architecture. Finally, Section VI presents the conclusions.

II. STATE OF THE ART

A. Internet of Things

The International Telecommunications Union defines the web of Things (IoT) because the "global infrastructure for the knowledge society, enabling advanced services by interconnecting (physical and virtual) things supported existing and evolving interoperable information and communication technologies" [4].

A wireless sensor network (WSN) may be a network composed by autonomous devices that uses sensors to watch environmental or physical conditions like vibration, temperature, sound, pressure or motion at different locations. Wireless sensor networks are now utilized in many application areas (e.g., smart cities, eHealth, Industry 4.0) [5]. The WSN is made of nodes, where each node is connected to at least one (or sometimes) several sensors, which give the info for IoT applications. The topology of the WSNs can vary from an easy star network to a complicated multi-hop wireless mesh network. Gateways are simplifying the IoT networks by supporting the connection of various network devices and by consolidating and mitigating the good variety and variety of disparate data coming from different network devices. an easy IoT gateway organizes and packetizes the info for transport over the web . it's also liable for distributing data back to finish points in applications where two-way communications are advantageous or required.

B. Software-Defined Networking

Software-Defined Networking (SDN) may be a specification that permits the programming of the network through clearly defined interfaces. SDN allows the likelihood of making new services and more efficient applications supported the interaction with networks traffic, network security implementation, or quality of service.
Given the heterogeneity of networks, it's challenging to coordinate and optimize the utilization of the heterogeneous network resources with the goal of satisfying as many tasks and services as possible. it's conjecture that the SDN paradigm may be a good candidate to unravel the resource management needs for network environments for multiple reasons [2]:

• SDN allows for a transparent separation between services within the control plane (that makes decisions about how traffic is managed) and therefore the data plane (actual mechanisms for forwarding traffic to desired destinations). The decoupling of the control plane from the forwarding plane encourages abstractions of low level network functionalities.
• Logically centralized view of the network, which allows to perform network optimization techniques. Redundancy and other mitigation failures are often applied so as to avoid single points of failure.

• Network programmability allows the dynamic and fast introduction of latest network services.

The OpenFlow protocol is an open source protocol that's a foundational element for building SDN solutions. This protocol allows network controllers to work out the flow paths across a network of switches thus enabling the straightforward traffic management through the separation of the control plane from the forwarding plane.
An OpenFlow Switch consists of 1 or more flow tables and a gaggle table, which perform packet lookups and forwarding, and an Open Flow channel to an external controller. The SDN controller manages the switch via the Open Flow protocol. Using this protocol, the controller can add, update, and delete flow entries, both reactively (in response to packets) and proactively [2]
The SDN controller may be a software entity that has exclusive control over an abstract set of knowledge plane resources. Several open source implementations of an SDN controller are available. Applications can run on top of SDN controller so as to supply advanced network services. SDN-enabled security applications are beginning to emerge as we discuss within the next subsection.

C. Security framework

Prior to the emergence of IoT, the adverse effects of threats were limited to theft of cash and intellectual properties. Now, the effect can cause loss of human life, hacking of critical infrastructures like electricity and atomic power grids, organizational productivity, and even national intelligence, so security becomes of the essence. The solutions like intrusion detection and prevention systems, encryption, and access management are the main solutions securing IoT. Many Network Intrusion Detection Systems (NIDSs) are developed by researchers and practitioners [6][7][8][9].

D. Anomaly Detection Solutions

The authors in [10] provide a classification of network anomaly methods. Several approaches are discussed:
a) Statistical methods and systems,
b) Classification-based methods and systems,
c) Clustering and Outlier-based methods and systems,
d) Soft computing methods and systems,
e) Knowledge-based methods and systems.

Focusing on statistical methods, an anomaly (from the statistical perspective) behavior is being considered partially or entirely inappropriate thanks to the very fact that's not generated by the assumed stochastic model. Statistical methods usually fit a model to the given data then apply a statistical inference test to work out if an unseen instance belongs to the present model. Instances that have a coffee probability to be generated from the learnt model, supported the applied test statistic, are declared anomalies.

E. Anomaly mitigation strategies

Two are the foremost important mitigation strategies. the primary focuses on limiting the effect of an anomaly to the network bandwidth, while the second tries to completely minimize the impact of an anomaly, and it are often understood as a corner case of the primary strategy.

1) Rate Limiting

In computer networks, rate limiting is employed to regulate the speed of traffic sent or received by a network interface controller. It are often induced by the network protocol stack of the sender and also by the network scheduler of any router along the way. the speed limiting (or traffic shaping) is that the regulation of the speed at which flows are allowed to inject packets into the network and is therefore one among the cornerstones of any QoS architecture. OpenFlow 1.3 allows different rate limiting policies once a particular rate has been reached: a) Drop, (discard) the packet, which may be wont to define a rate limiter band; b) Dscp, which decreases the drop precedence of the DSCP field within the IP header of the packet. are often wont to define an easy DiffServ police.

2)Flow interruption

In order to try to to not allow any traffic of a suspicious flow, the flow rule are often directly far away

from the SDN controller, which may be considered a specific case of the previous technique.

III. SDN-ENABLED SECURITY FRAMEWORK

Our proposed IoT security architecture consists of three main blocks, which are show in Fig. 1: an SDN/NFV edge node, an SDN controller and an E2E security Application.

The SDN/NFV Edge Node may be a distributed computing infrastructure (or fog computing) during which some services are handled at the network edge. The goal of this fog node is to enhance efficiency and reduce the quantity of knowledge that must be transported to the cloud for processing , analysis and storage. this is often often finished improving the efficiency of the network, but it's going to even be implemented for security and compliance reasons. This Fog node has the power to act as:
a) An Open Flow-enabled switch so as to modify the various IoT gateways that are connected. Moreover connectivity to aggregation and transport networks is additionally provided.

b) Virtual Machines running different services, like an IoT database, where the measurements of the various sensors are stored for local processing.

The SDN controller shall be ready to support Open Flow control of flow tables, meters and actions. a transparent North Bound Interface (NBI) of the SDN controller shall be described so as to ease the mixing of upper level applications.
The End-to-end security Application are going to be liable for monitoring of the present flows, and thru different anomaly detection mechanism it'll be ready to identify malicious flows. Finally, this application will apply the specified security policies with reference to the detected anomalies so as to mitigate them. It consists of three main modules shown in Fig. 1: (a) the Collector, (b) the Anomaly Detection and (c) the Anomaly Mitigation.

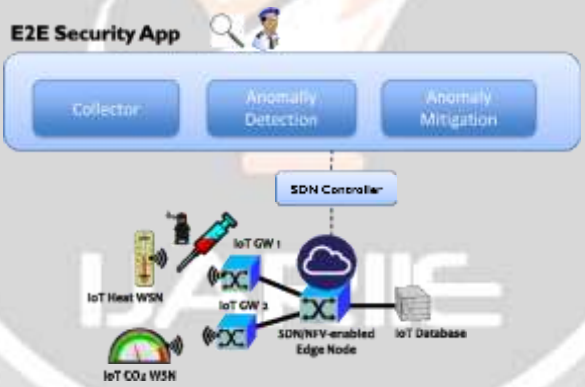


Fig. 1. SDN-enabled Security Framework

Collector
The Collector module is responsible for gathering the data, a prerequisite in order to perform flow-based anomaly detection. This module collects flow information and periodically report them to the Anomaly Detection module. If we focus on the available information from the SDN controller, the different per flow counters are obtained. From this information we can estimate the Packets per second per flow; and the bytes per second per flow. Anomalies in the flows, such as for example intrusion detection, or malfunctions are detected from the statistics from the gathered data.

Anomaly Detection Module
The data generated by the previous module are subsequently fed to the Anomaly Detection module at periodic time intervals, thus creating discrete time windows. For every time window this module inspects all the flow entries, exposing any flow-related network anomaly and identifying a potential attacker or the victim of the attack. Moreover, it performs successfully not only in identifying the attack pattern (e.g., flood), but also the victim (i.e., host under attack). Once an anomaly is detected in the network, the proposed algorithm inspects and correlates specific network metrics by identifying the attack and exposing all related information to the

Anomaly Mitigation module.

In particular, in the context of abuse and network intrusion detection, the interesting objects are often unexpected bursts in activity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular unsupervised methods) will fail on such data, unless it has been aggregated appropriately.

The most common measures of dispersion for continuous data are the variance and standard deviation. Both of them describe how much the individual values in a data set vary from the mean or average value. The variance and standard deviation are calculated slightly differently depending on whether a population or a sample is being studied, but basically the variance is the average of the squared deviation from the mean, and the standard deviation is the square root of the variance.

For the detection of the anomalies it has been envisaged an algorithm based on the data variance gathered from any sensor device in the network. The formula presented below is the deviation calculated for the n data samples (xi) provided and its mean by the sensors

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2} \qquad (1)$$

The malware detection for the data processed by the sensors and is conditioned by the following formula, where n is a configurable parameter:

computed the error rate of the algorithm, including both false positive and negative results.

The simulations were carried out for N_SIGMA= 10 and 1000 samples. For each WINDOW length the algorithm was simulated 3 times and each time the error was tracked, afterwards it has been calculated the mean value of Error (%). As a first approach it is observed that when incrementing the window of simulation from 5 to 10 the error is decreased (see Fig. 2). This is due to the fact that a small window size, doesn't leave time to properly measure the flow standard deviation. Decisions made by this incorrect measurement of standard.

A. Anomaly Mitigation Module

incremented. The optimal window size result is an error rate in anomaly detection of 3,9%.

The Anomaly Mitigation module aims to neutralize identified attacks, by inserting flow meters in the flow table of the Open Flow switch (or removing existing flows) in order to block/mitigate the desired malicious traffic. These flow entries have higher priority than any other in the flow table.

SIMULATION RESULTS

The IoT security algorithm has been developed by using Python programming language. A conformant flow is created with the following properties: a) Using the base of 20 packets/second, which shall match for example the readings of 20 sensors each second, we introduce a small uniform variance; and b) The number of bytes per second is linearly obtained from the packets per second (using a standard 100 bytes per packet). Instead, a bad flow is created with a probability of 10%. A bad flow has different properties: a) The number of packets per second are duplicated (in comparison with a conformant flow); and b) The packet size is also 50% increased.

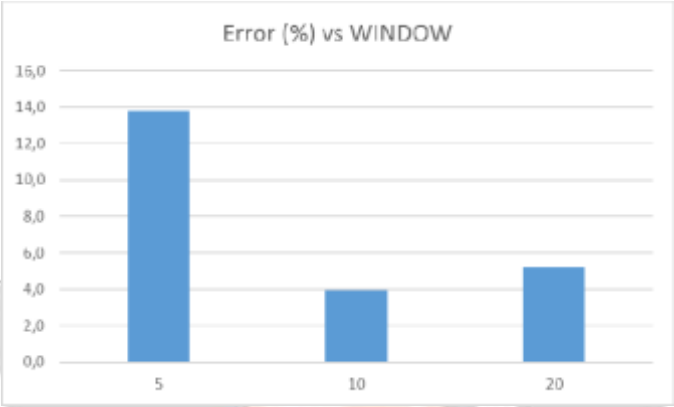


Fig. 2. Simulation results for error rate in anomaly detection (%) depending on window size

It is important to mention that the simulations were carried out for 5 simulated parameters N_SIGMA (n) and 3 simulated parameters of WINDOW for both types of the simulated traffic, packets/s and bytes/s. Firstly we selected an optimal N_SIGMA of 10, after several N_SIGMA variations. We

EXPERIMENTAL VALIDATION

In order to carry out the experimental part of this work two official platforms developed by the CTTC´s research staff has been interconnected, the IoTWorld® Testbed and the ADRENALINE Testbed® (Fig 3). The ADRENALINE

cloud/fog computing platform and transport network offer an integrated cloud and network orchestration which is capable of creating VMs and establishing End-to-End (E2E) paths through heterogeneous networks considering multi-domain SDN orchestration. The deployed architecture allows providing E2E connectivity services of VMs located in different network locations, interconnected though multi-layer multi-domain networks considering heterogeneous SDN/OpenFlow (OF) and General Multiprotocol Label Switches (MPLS)/Path Computation Element control planes [11].

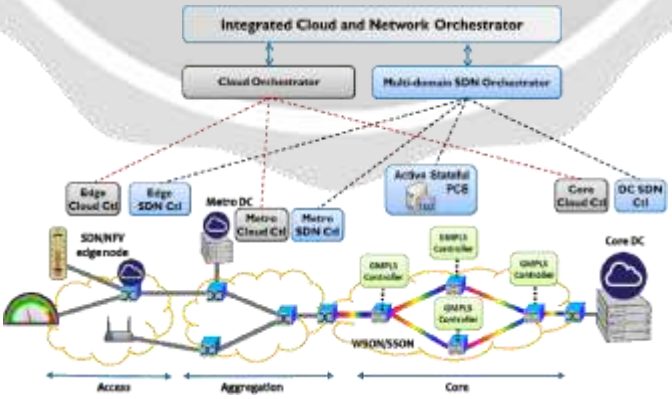


Fig. 3. Overall ADRENALINE & IoTWORLD Testbed

The IoTWORLD®, is an End-to-End platform for the Internet of Things. The main focus is on Wireless Communications systems and data analytics. The platform has been deployed in two different neighbour buildings: in a laboratory, in an isolated room, and in a real office environment. Different sensors and actuators are connected to a set of gateways, either with a direct connection or via multiple hops. These gateways are then connected to the Internet, providing the capability to retrieve and store data in the cloud, among other

functionalities, such as data fusion, compression and analytics.

For the sake of the experiments the IoT platform consists of a temperature and a dioxide carbon (CO2) sensors, 2 Wireless (802.15.4 compliance) nodes and 2 gateways that gather the data generated by the sensors, one SDN NFV Edge Node to control the flows of data coming from the sensors and a database where the data is stored. The SDN NFV Edge Node network element is able to provide computing, storage and networking services, acting as a cloud edge node (fog node) and has been introduced in [12].

Fig. 3 presents the considered system architecture which includes the proposed SDN/NFV-enabled edge node. The proposed node consists of an OF-enabled switch, compute and storage resources. The switch is controlled via an edge SDN controller, while the computation and storage resources are controlled with an edge cloud/fog controller. On top of the proposed architecture, the Integrated Cloud/Fog and Network Orchestrator is responsible for handling a VM and network connectivity requests, which are processed through the Cloud/Fog and Multi-domain SDN orchestrators.

We have extended OpenStack Mitaka in order to provide control for distributed availability zones, we refer as Cloud Orchestrator. For Multi-domain SDN Orchestrator we have developed an Application-Based Network Operations architecture, which has been previously demonstrated in [11]. It allows the provisioning of E2E connectivity services through different underlying SDN-controlled network domains. In this paper, we focus on describing the access network of the tested, which is controlled by with the Integrated Cloud and Network Orchestrator, which is able to deploy applications and services on top of the testbed infrastructure.

The proposed E2E security application runs on top of the SDN/NFV-enabled edge node (see Fig. 1). The proposed SDN/NFV edge node has been developed using an Intel NUC NUC5i5RYH, with 16Gb RAM and 120 Gb SSD. Several USB to Ethernet port converters have been included in order to extent the node switching capabilities.

The wireless nodes are Z1 motes by Zolertia, which feature a 16-bit RISC CPU @16MHz, an 8KB RAM and a 92 KB flash memory. They also include the CC2420 transceiver, which is IEEE 802.15.4 compliant. These nodes support ContikiOS, an open-source operating system for the IoT, which connects tiny, low-cost, low-power microcontrollers to the Internet and supports Ipv6 through 6LowPAN. It is worth noting that each mote can operate as either a source or a sink node. Finally, the IoT gateways are connected to 2 sink nodes via USB. IoT gateways have been implemented using a Raspberry Pi 2, using Raspbian, based on Debian. The IoT gateway reads the measurements in the USB port and it retransmits them to a processing software, which is run in a VM running at the edge node.

In Fig. 4, the overall system from the networking perspective is depicted. First, two IoT gateways are interconnected towards a virtual software switch (SW1), running on the edge node. This external virtual software switch is interconnected towards an internal virtual software switch (SW2) into with the virtual machines are connected. Finally, the IoT virtual machine (test_iot) is connected to this second switch.
Computed the error rate of the algorithm, including both false positive and negative results.

The simulations were carried out for N_SIGMA= 10 and 1000 samples. For each WINDOW length the algorithm was simulated 3 times and each time the error was tracked, afterwards it has been calculated the mean value of Error (%). As a first approach it is observed that when incrementing the window of simulation from 5 to 10 the error is decreased (see Fig. 2). This is due to the fact that a small window size, doesn't leave time to properly measure the flow standard deviation. Decisions made by this incorrect measurement of standard deviation will be error prone. If we increment the window length from 10 to 20, the error rate in anomaly detection is

$$x_i \square x \square n \square\square$$

Anomaly Mitigation Module

(2)incremented. The optimal window size result is an error rate in anomaly detection of 3,9%.
The Anomaly Mitigation module aims to neutralize identified attacks, by inserting flow meters in the flow table of the Open Flow switch (or removing existing flows) in order to block/mitigate the desired malicious traffic. These flow entries have higher priority than any other in the flow table.

SIMULATION RESULTS

The IoT security algorithm has been developed by using Python programming language. A conformant flow is created with the following properties: a) Using the base of 20 packets/second, which shall match for example the readings of 20 sensors each second, we introduce a small uniform variance; and b) The number of bytes per second is linearly obtained from the packets per second (using a standard 100 bytes per packet). Instead, a bad flow is created with a probability of 10%. A bad flow has different properties: a) The number of packets per second are duplicated (in comparison with a conformant flow); and b) The packet size is also 50% increased.

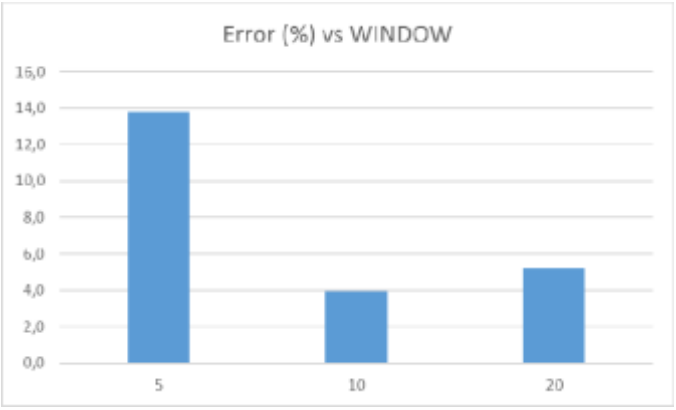


Fig. 2. Simulation results for error rate in anomaly detection (%) depending on window size

It is important to mention that the simulations were carried out for 5 simulated parameters N_SIGMA (n) and 3 simulated parameters of WINDOW for both types of the simulated traffic, packets/s and bytes/s. Firstly we selected an optimal N_SIGMA of 10, after several N_SIGMA variations.

EXPERIMENTAL VALIDATION

In order to carry out the experimental part of this work two official platforms developed by the CTTC´s research staff has been interconnected, the IoTWorld® Testbed and the ADRENALINE Testbed® (Fig 3). The ADRENALINE

cloud/fog computing platform and transport network offer an integrated cloud and network orchestration which is capable of creating VMs and establishing End-to-End (E2E) paths through heterogeneous networks considering multi-domain SDN orchestration. The deployed architecture allows providing E2E connectivity services of VMs located in different network locations, interconnected though multi-layer multi-domain networks considering heterogeneous SDN/OpenFlow (OF) and General Multiprotocol Label Switches (MPLS)/Path Computation Element control planes [11].

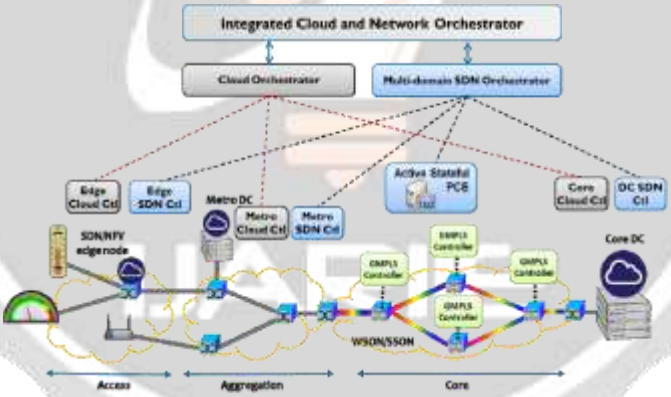


Fig. 3. Overall ADRENALINE & IoTWORLD Testbed

For the sake of the experiments the IoT platform consists of a temperature and a dioxide carbon (CO2) sensors, 2 Wireless (802.15.4 compliance) nodes and 2 gateways that gather the data generated by the sensors, one SDN NFV Edge Node to control the flows of data coming from the sensors and a database where the data is stored. The SDN NFV Edge Node network element is able to provide computing, storage and networking services, acting as a cloud edge node (fog node) and has been introduced in [12].

Fig. 3 presents the considered system architecture which includes the proposed SDN/NFV-enabled edge node. The proposed node consists of an OF-enabled switch, compute and storage resources. The switch is controlled via an edge SDN controller, while the computation and storage resources are controlled with an edge cloud/fog controller. On top of the proposed architecture, the Integrated Cloud/Fog and Network Orchestrator is responsible for handling a VM and network connectivity requests, which are processed through the Cloud/Fog and Multi-domain SDN orchestrators.

We have extended OpenStack Mitaka in order to provide control for distributed availability zones, we refer as Cloud Orchestrator. For Multi-domain SDN Orchestrator we have developed an Application-Based Network Operations architecture, which has been previously demonstrated in [11]. It allows the provisioning of E2E connectivity servicesthrough different underlying SDN-controlled network domains. In this paper, we focus on

describing the access network of the testbed, which is controlled by with the Integrated Cloud and Network Orchestrator, which is able to deploy applications and services on top of the testbed infrastructure.
The proposed E2E security application runs on top of the SDN/NFV-enabled edge node (see Fig. 1). The proposed SDN/NFV edge node has been developed using an Intel NUC NUC5i5RYH, with 16Gb RAM and 120 Gb SSD. Several USB to Ethernet port converters have been included in order to extent the node switching capabilities.
The wireless nodes are Z1 motes by Zolertia, which feature a 16-bit RISC CPU @16MHz, an 8KB RAM and a 92 KB flash memory. They also include the CC2420 transceiver, which is IEEE 802.15.4 compliant. These nodes support ContikiOS, an open-source operating system for the IoT, which connects tiny, low-cost, low-power microcontrollers to the Internet and supports Ipv6 through 6LowPAN. It is worth noting that each mote can operate as either a source or a sink node. Finally, the IoT gateways are connected to 2 sink nodes via USB. IoT gateways have been implemented using a Raspberry Pi 2, using Raspbian, based on Debian. The IoT gateway reads the measurements in the USB port and it retransmits them to a processing software, which is run in a VM running at the edge node.
In Fig. 4, the overall system from the networking perspective is depicted. First, two IoT gateways are interconnected towards a virtual software switch (SW1), running on the edge node. This external virtual software switch is interconnected towards an internal virtual software switch (SW2) into with the virtual machines are connected. Finally, the IoT virtual machine (test_iot) is connected to this second switch.
The traffic is routed from IoT Gateway 1 with MAC address b8:27:eb:4f:67:5b and IP address 10.1.14.7 through the source: OpenFlow switch with the following MAC address 00:00:80:3f:5d:08:2b:72 and interface connected to port 1, with the following IP address 10.1.7.48 and the destination: a switch with the following MAC address 00:00:6a:5a:fb:3b:66:41, this switch is in charge of managing the traffic of the IoT SQL Database through an interface of the controller, number 41, with the following IP address 192.168.30.30.

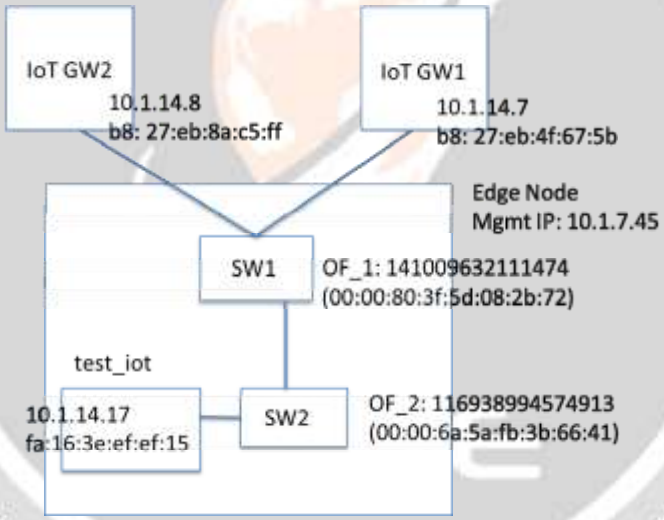
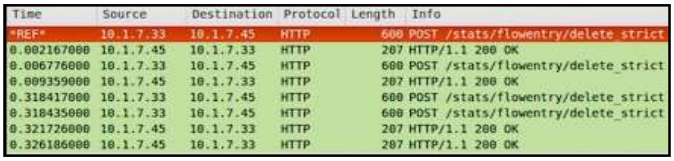


Fig. 4. Overall system configuration

The traffic is routed from Gateway 2 with MAC address b8:27:eb:8a:c5:ff and IP address 10.1.14.8 through the source, OpenFlow switch with the following MAC address 00:00:80:3f:5d:08:2b:72 and interface connected to port 3, with the following IP address 10.1.7.49 and the destination: a switch with the following MAC address 00:00:6a:5a:fb:3b:66:41 , this switch is in charged to manage the traffic of the IoT SQL Data Base through an interface of the controller, number 41, with the following IP address 192.168.30.30.
The Collector module is responsible for data gathering, a prerequisite in order to perform flow-based anomaly detection. This module collects flow information and periodically exports them to the Anomaly Detection module. The flows are recorded as per packets per second and bytes per second as presented in the traffic frame presented below.
The data produced by the Data Collector are subsequently fed to the Anomaly Detection module at fixed time intervals, thus creating discrete time windows. For every time-window this module inspects all flow entries, exposing any flow-related network anomaly and identifying a potential attacker or the victim of the attack. Moreover, it performs successfully not only in identifying the attack pattern, but also the attacker or the victim (i.e. host under attack). As soon as an anomaly is detected in the network, our algorithm inspects and correlates specific network metrics in order to identify the attack and forward all related information to the Anomaly Mitigation module. The used algorithms are the previously presented in Section III.
The Anomaly Mitigation module aims to neutralize identified attacks, inserting flow-entries in the flow table of

the Open Flow switch (or modify existing ones) in order to block the desired malicious traffic.
A widely used and easy to be implemented method for the flow mitigation is the flow limiting (rate limit) in the OF Switch. In this demonstration, the flow interruption mechanism has been used. Fig. 5 shows the message exchange using a Wireshark capture. It can be seen the flow traffic eliminated from the switch OF_1, IP address 10.1.7.45.



Time	Source	Destination	Protocol	Length	Info
REF	10.1.7.33	10.1.7.45	HTTP	600	POST /stats/flowentry/delete_strict
0.002167000	10.1.7.45	10.1.7.33	HTTP	207	HTTP/1.1 200 OK
0.006776000	10.1.7.33	10.1.7.45	HTTP	600	POST /stats/flowentry/delete_strict
0.009359000	10.1.7.45	10.1.7.33	HTTP	207	HTTP/1.1 200 OK
0.318417000	10.1.7.33	10.1.7.45	HTTP	600	POST /stats/flowentry/delete_strict
0.318435000	10.1.7.33	10.1.7.45	HTTP	600	POST /stats/flowentry/delete_strict
0.321726000	10.1.7.45	10.1.7.33	HTTP	207	HTTP/1.1 200 OK
0.326186000	10.1.7.45	10.1.7.33	HTTP	207	HTTP/1.1 200 OK

Fig. 5. Wireshak: Flow limiting in the OF Switch

CONCLUSIONS

We have demonstrated the feasibility to use an SDN- enabled security framework and we have proposed a security architecture for IoT devices, which is based on the principles of SDN. A simple algorithm has been implemented to analyze the feasibility of statistical analysis for anomaly detection. We have introduced the proposed SDN security application for IoT by interconnecting the ADRENALINE and IOTWORLD testbeds, running on top of an SDN/NFV-enabled edge node. Finally, we have demonstrated the flow interruption anomaly mitigation technique.
Further research on security for IoT needs to be performed, but using the powerful framework on SDN, has been demonstrated as a useful weapon against security threads.

ACKNOWLEDGMENT

The research leading to these results has received funding from Spanish MINECO projects DESTELLO (TEC2015-69256-R) and CellFive (TEC2014-60130-P) and the Generalitat de Catalunya under grant (2014SGR 1551).

REFERENCES

[1] Bonomi, Flavio, et al. "Fog computing and its role in the internet of things." Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012.

[2] Nadeau, Thomas D., and Ken Gray. SDN: software defined networks. O'Reilly Media, Inc., (2013).
[3] Telefonica, “Trend Report. Insecurity in the Internet of Things”, pp 3- 10, (2015)
[4] ITU-T Y.2060 (06/2012), Overview of the Internet of things.

[5] Dinesh Kumar Gupta , “A Review on Wireless Sensors Networks”,
[6] Network and Complex Systems , ISSN 2224-610x

[7] Li, Bingdong, et al. "A survey of network flow applications." Pp 567- 581, Journal of Network and Computer Applications 36.2 (2013)