IDIT-SDN: Intrusion Detection Framework for Software-defined Wireless Sensor Networks

Gustavo A. Nunez Segura[1], **Arsenia Chorti**[2], **Cíntia Borges Margi**[3]

[1]Escuela de Ingeniería Eléctrica – Universidad de Costa Rica

[2]ETIS UMR8051, CY Université, ENSEA, CNRS, F-95000, Cergy, France

[3]Escola Politécnica – Universidade de São Paulo, São Paulo, SP, Brazil

gustavoalonso.nunez@ucr.ac.cr, arsenia.chorti@ensea.fr, cintia@usp.br

Abstract. *Software-Defined Networking has been used to leverage security solutions for wireless sensor networks. However, this paradigm turns networks vulnerable to distributed denial of service attacks.* IDIT-SDN *is a tool for Software-defined Wireless Sensor Networks devised for DoS and DDoS attacks simulation and detection. This tool provides a framework for anomaly detection and a communication protocol to share security wise information from the sensor network to the controller. We demonstrate its use by showing a cooperative DDoS attack detection and attacker identification application based on distributed (every node) and centralized (controller) anomaly detection.*

1. Introduction

Wireless Sensor Networks (WSN) is a data collection technology used for Internet of Things (IoT) and Industry 4.0 applications. We can find WSNs deployed in different places, such as: shopping centers, hospitals, cities, or even inside the forest, deserts or water masses. Moreover, the type of data collected might change according to the application. A WSN can be deployed inside a building for environmental monitoring while in the same building someone could be using a wearable for blood pressure control. Perhaps losing a temperature sample packet is not critical but for the blood pressure control it is. This means, there is a wide range of service requirements a WSN should fulfill.

Security is a challenging requirement for WSNs. An specific application that collects sensible or private data using resource constrained devices may be deployed in a hostile location. In these conditions, the network is vulnerable to all kind of attacks and have few resources to detect and mitigate them.

Software-Defined Networking (SDN) [McKeown et al. 2008] has been explored to improve security in WSNs. SDN proposes to move control decisions from network devices to dedicated hardware, commonly known as SDN Controller. The controller receives or collects data from network devices to then configure forwarding rules on them. This view of the network is instrumental to detect and mitigate attacks, however, centralization of control decisions turns SDN networks prone to DoS and DDoS attacks [Kreutz et al. 2015], since the whole network operation depends on the controller.

One strategy to reduce this vulnerability is to implement part of the attack detection in the network devices [Naous et al. 2009, Ahmad et al. 2015]. Doing this, we aim

to detect the attack before it reaches the controller. To attain this objective, we developed *IDIT-SDN*, a security framework for Software-Defined Wireless Sensor Networks (SDWSN). *IDIT-SDN* includes a security and a management module in every node in the network. The management module collects information to calculate performance metrics and the security module uses these metrics to detect anomalies in its behavior. Then, the security modules are able to share this information with the SDN Controller to determine if the network is under attack.

In this paper, we explain the architecture of the framework, the communication protocol, the detection algorithm and its implementation. Then, we show its basic configuration and a demonstration of its use.

2. Architecture

IDIT-SDN enables the cooperation between the sensing layer and the control layer. This cooperation aims to improve DoS and DDoS detection in SDWSN.

A security and a management applications run on the SDN controller. The management application receives data from the management application in every network device to calculate global performance metrics, such as delivery rate and control overhead. The security application uses the metrics calculated by the management application to perform statistical analysis and detect anomalies on the network behavior. From this point, we will use the names *CSec* and *CMan* for the security and management applications in the SDN controller, respectively. Likewise, a security and a management applications run on every network device. The operation is similar to the case in the SDN controller: the security application uses the metrics from the management application to detect anomalies. However, in this case the data source of the management application is the network device itself. Some examples of metrics are: number of packets transmitted and processing resources usage (time). From this point, we use the names *NSec* and *NMan* referring to the security and management applications in every network device, respectively.

Individually, every network device and the SDN controller are able to detect anomalies that could relate to DoSs or DDoS attack. Then, we propose that CSec receives information from the *NSecs*, such as notifications of anomaly detection and the metric, or metrics, where an anomaly was detected. We believe the main contribution of this tool is the implementation of security policies that take advantage of centralized and distributed information in the SDN controller to improve DoS and DDoS attacks detection. The architecture of *IDIT-SDN* is depicted in Figure 1.

3. IDIT-SDN implementation

IDIT-SDN is currently implemented on IT-SDN [Alves et al. 2017] version 0.41 [1]. IT-SDN is an open source SDWSN developing framework, implemented as a Contiki OS network driver. This framework is composed by three main protocols: Southbound protocol, Neighbor Discovery protocol and Controller Discovery protocol. The Southbound protocol handles the communication between the SDN controller and the network device, which includes the definition of the packet format, the processing of packets and the operation workflow. The neighbor discovery protocol manages neighborhood information

[1] Available at `https://sites.google.com/usp.br/cintia/it-sdn`

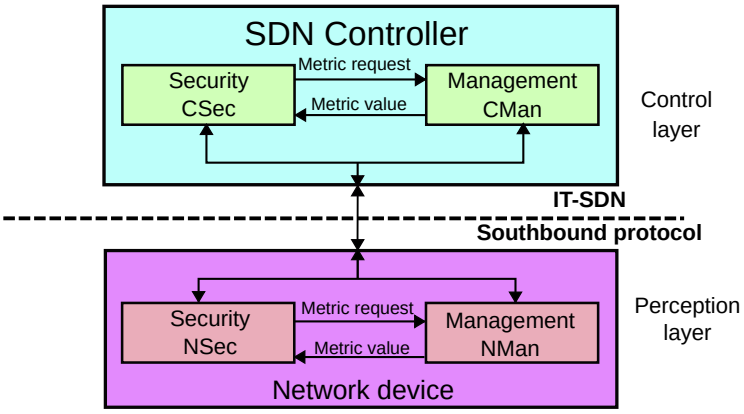


Figure 1. IDIT-SDN architecture

and controller discovery protocol is in charge of obtaining a route to reach the controller. The operation of these three protocols is orchestrated by an event based SDN core.

IT-SDN includes a controller software implementation. This software collects and stores topology information to construct the topology graph and takes routing decisions based on Djikstra algorithm. One network device works as a bridge between the controller and the WSN. A serial protocol to communicate the controller and this bridge is also part of the implementation provided. IT-SDN performance was evaluated in topologies with up to 289 nodes and it is similar to RPL's performance [Alves et al. 2019].

The *CSec* and *CMan* are applications that reside in the SDN Controller while the *NSec* and *NMan* are applications that reside in the Network device. However, the IT-SDN control plane and Southbound protocol remains without changes.

3.1. Anomaly detection

Intrusion detection is classified in two categories: signature-based and anomaly-based. Signature-based detection uses pattern matching to identify known attacks but is not effective to zero-day attacks. Anomaly-based detection is meant to identify abnormal behaviors by using statistical methods or machine learning, regardless they match a pattern or not.

A symptom of a DoS and DDoS attacks is an anomaly behavior on packets traffic. We implemented a real time anomaly detector based on Change Point (CP) analysis, following the work proposed in [Skaperas et al. 2019]. Their proposal is composed of an off-line phase, an on-line phase and a trend indicator. The off-line phase is a training period employed to configure the on-line phase and the trend indicator determines the direction of the change. Since the anomaly detector should fit in resource-constrained devices, we decided to implement only the on-line phase and configure the parameters according to our requirements: accuracy versus detection speed. The low-complexity (i.e. $O(N \log N)$, where N is the length of the time series), success rate and access to the original code were the main reasons to choose it.

3.2. *NSec* and *CSec*

NSec was implemented as a Contiki process thread that initiates after all the communication processes are up. The anomaly detection algorithm iterates between monitoring and

detection routines. The monitoring routine is meant to obtain statistical information of the network devices behavior while the detection routine is meant to detect anomalies. The monitoring routine follows the next two steps: i), a time series with samples of the metric of interest is constructed; ii) after M samples, the cumulative sum and variance of the time series are calculated and stored. The value of the metric is provided by the *NMan*.

The detection routine starts immediately after the monitoring routine. The detection routine continues the metric sampling and, after every new sample, executes an hypothesis test. If a CP is detected, an alarm is triggered and a message is sent to the *CSec*. On the other hand, if after M samples no CP is detected, the algorithm goes back to the monitoring routine. Since there are memory restrictions on the device, the first D samples of the current monitoring time series are discarded and the D samples collected during the detection routine are copied to the monitoring time series. Then, a new cumulative sum and variance are calculated.

The *CSec* executes the same CP detection algorithm explained for the *NSec*. The information to construct the time series is provided by the *CMan*. Since the *CSec* is supposed to run in a server without resource constraints, we are able to implement multiple detectors, where each detector could monitor one metric. Monitoring multiple metrics enables the implementation of policies to determine the type of the attack [Segura et al. 2022] or confirmation of attack based on multiple CP.

Other security policies or detection algorithms could be implemented using the information received from the NSecs. One example is attacker identification. When the *CSec* receives an alarm from a *NSec*, it is able to ask to the SDN controller about the neighborhood of this node. If the *CSec* receives more alarms from the same neighborhood, it is possible that the attacker be within this area. We have implemented an attacker identification algorithm that works in case of new-flow-based attacks. In this algorithm, the network device has to determine a suspect before sending the alarm. To solve this problem, the network device is aware of packets with flow identifiers that are not in its flow table and required a flow request to route it. The network device stores the address of the source of the packet in a buffer and, when an anomaly is detected, it checks the buffer and counts the repetition of each address. The address with more repetitions is the one reported to the *CSec*. The *CSec* can now take a decision based on the number of alarms received reporting the same address [Segura et al. 2022].

3.3. Communication

The *NSec* is able to inform to the *CSec* when an anomaly is detected through the communication protocol for management communication proposed in [Luz et al. 2019]. This protocol has two packets: Request Packet and Monitoring Packet. The Request Packet allows a centralized management to request information to a specific network device. The network device replies using the Monitoring Packet. The protocol supports asynchronous communication for periodic monitoring, thus network devices are able to send management information to the centralized application even without receiving a request.

Every network device is able to communicate an alarm to the *CSec* by sending a packet with flow *1*, which is the flow identifier IT-SDN uses to reach the controller. The packet should be identified as `SDN_PACKET_MNGT_NODE_DATA` to be routed to the *CMan* application. Then, the *CMan* message handler checks the message header. If the

header matches with `SDN_SEC_CP_DETECT_ALARM`, the message handler forwards the information to the *CSec*.

Figure 2 depicts the Monitoring Packet format. The header contains packet type (1 byte), TTL (1 byte), sequence number (1 byte), source address (2, 6 or 8 bytes) and one byte reserved for compatibility with Rime packets in the Contiki communication stack. Flow ID contains the flow identifier (2 bytes). Management metrics are two bytes to inform the metrics values contained in the message. Each bit is a flag that represents one metric; if one bit is *1*, the metric related to this bit is contained in the message. Thus, the protocol supports up to 16 metrics. In the current version of *IDIT-SDN*, we use the eighth bit to send an alarm to the *CSec*. In this case the suspect address is contained in the metric value.

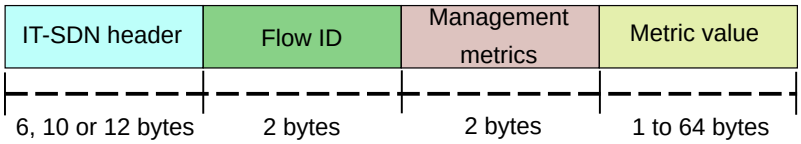


Figure 2. Monitoring Packet format

4. Configuration

IDIT-SDN has been used in networks up to 225 nodes, obtaining DDoS attack detection results above 95% and intrusion identification above 93% [Segura 2021]. However, our intention is to show its use as a research tool for security in WSN and IoT environments. For this purpose, we will explain the basic configuration of the *NSec* and *CSec*.

The ***NSec*** was implemented as a Contiki process thread named *security_node_process*. Its code is located in the file *sdn-node-security-module.c* inside the folder *sdn-common*. The number of samples for the monitoring window (`m_win`) and detection window (`d_win`) should be set in this file. Results about the impact of these parameters on the detection performance are presented at [Segura et al. 2022]. Also, it is worth to consider memory usage in case of resource constrained devices.

The sampling period is configurable. It was implemented assuming a uniform sampling during the whole simulation, thus, a flag in the makefile (*Makefile_enabled_node*) could be activated to set this time. The options implemented are: `SONETWSEC` for one hundred and twenty seconds, `SSIXTYSEC` for sixty seconds, `STHIRSEC` for thirty seconds and `STENSEC` for ten seconds.

The CP detection algorithm has two configurable parameters: sensitivity (`g`) and critical value (`ca`). However, the values of these parameters are restricted. The sensitivity can be set to: 0, 0.15, 0.25, 0.35, 0.45, 0.49. The critical value depends on another parameter, which is not configurable for *NSec*, but all the possible values are available in the file *sdn-security-module.c*. We use $ca = 2.82$ by default. The discussion about these parameters is presented in [Segura et al. 2022].

In the current version of *IDIT-SDN*, *NSec* is able to monitor just one metric at a time. By default, it uses the transmitting time to detect anomalies. This metric is the time the radio module remains on transmitting. The detection performance of DoS and DDoS attacks using this metric can be found it [Segura et al. 2022].

The ***CSec*** was implemented as an application that resides in the SDN controller. The main code is located in the file *sdn-security-module.c* inside the folder *controller-server*.

The code is split in two parts: one part for the centralized detection and the other part to receive and process the alarms from the *NSecs*. The centralized detection is composed by two CP detectors, one detector to analyze control packets overhead and the other one to analyze data packets delivery rate behavior. To enable the operation of the detectors, you should set the flags `CTRL_OV_DETECT` and `DATA_DR_DETECT` in the makefiles and in the file *controller-pc.pro*.

The sensor nodes should be configured to provide the information required to calculate the control overhead and data packets delivery rate metrics in the *CMan*. For this, there are four flags that define the periodicity of this report. The options implemented are: `MONETWSEC` for one hundred and twenty seconds, `MSIXTYSEC` for sixty seconds, `MTHIRSEC` for thirty seconds and `MTENSEC` for ten seconds. The time set for the sensor nodes should be the same set for the *CMan* in the file *controller-pc.pro*. However, there is a parameter that should be manually configured. In the current version, *IDIT-SDN* is not able to synchronize the report of data packets sent by every sensor node and the number of data packets received by the sink, thus, the number of data packets expected should be set in the file *sdn-metrics-cal.c* located in the folder *controller-server*.

The parameters `S_SKIP`, `T_WIN`, `GAMMA` and `S_VAL` are configurable. `S_SKIP` is a number of samples ignored by the CP analysis. This is meant to avoid the transitory behavior during the initial configuration of the network. `T_WIN` is the size of the monitoring window. `GAMMA` (γ) and `S_VAL` (α) are the parameters to calculate the critical value. Please check references [Segura et al. 2022] to deepen this topic.

In the case a detector raises an alarm, the algorithm checks which metric triggered the detector and classifies the possible attack. We have tested the attack identification for two type of DDoS attacks: new-flow-based and neighborhood-information-based. Our results showed that new-flow-based attacks are detected first by monitoring the number of control packets and that neighborhood-information-based attack are detected first by data packets delivery rate [Segura et al. 2022].

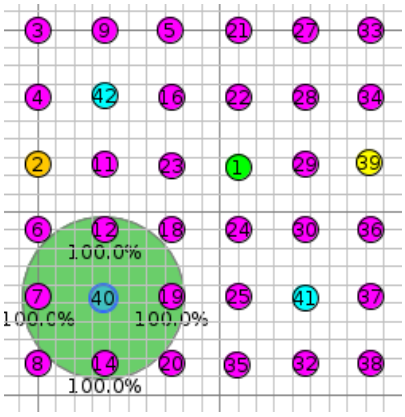
The module that receives the *NSecs* alarms provides a list of suspects and a list of attackers, based on the hypothesis that suspects with multiple indications are considered as attackers. The number of indications depends on the number of neighbors of this node and a threshold, which is configurable. This parameter is named `D_THRESHOLD` and is set by 0.5 as default.

4.1. Case of use: 36 nodes

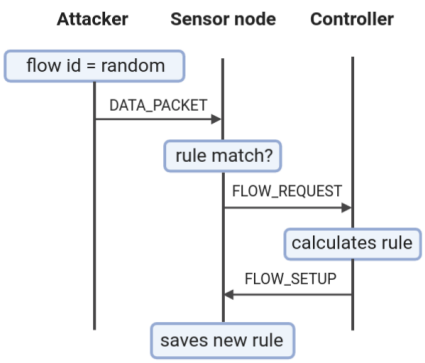
Figure 3a shows square grid topology of 36 nodes, composed of one SDN controller, two sinks and 33 sensor nodes. The *CSec* and *CMan* were configured to monitor control packets traffic and data packets delivery rate. The *NSec* and *NMan* were configured to monitor transmitting time. Three of these sensor nodes are programmed to behave as attackers and execute a new-flow-based attack variation named False Data Flow Forwarding (FDFF) [Segura et al. 2019]. In brief, the attackers fool their neighbors to flood the network with flow requests packets by sending data packets with unknown flow identifiers.

The attackers are identified with numbers 40, 41 and 42. The results expected are the CP detected by the *NSecs*, the CP detected by the *CSec* and the list of attackers identified.

[2]



(a) Topology with 36 nodes



(b) FDFF attack message exchange

Figure 3. Demonstration topology and attack message exchange

The simulation was executed using COOJA Simulator, which is part of Contiki. COOJA is capable of emulating WSN platforms. For these experiments, we used the sky mote (i.e. TelosB mote). This platform is equipped with a 8 MHz MSP430 microcontroller and the total footprint, considering IT-SDN stack, IDIT-SDN modules and a simple data monitoring application, is 48162 B of ROM and 9374 B of RAM. Table 1 summarizes the simulation parameters for this demonstration. The source code and user manual for *IDIT-SDN* is available at `https://github.com/gnunezucr/idit-sdn`, while the video demonstrations are available at `https://youtu.be/sYbSuFbqL2c` and `https://youtu.be/vFcMjfWg_UQ`.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and by the ELIOT project (ANR-18-CE40-0030 / FAPESP 2018/12579-7). Gustavo A. Nunez Segura is supported by Universidad de Costa Rica. Cintia B. Margi is supported by CNPq fellowship #311687/2022-9.

References

Ahmad, I., Namal, S., Ylianttila, M., and Gurtov, A. (2015). Security in software defined networks: A survey. *IEEE Communications Surveys & Tutorials*, 17(4):2317–2346.

Alves, R. C. A., Oliveira, D., Segura, G. N., and Margi, C. B. (2017). IT-SDN: Improved architecture for SDWSN. In *XXXV Simpósio Brasileiro de Redes de Computadores*. Available at `https://sites.google.com/usp.br/cintia/it-sdn`.

Alves, R. C. A., Oliveira, D. A. G., Nunez Segura, G. A., and Margi, C. B. (2019). The Cost of Software-Defining Things: A Scalability Study of Software-Defined Sensor Networks. *IEEE Access*, 7:115093–115108.

[2]The firmware used to implement the FDFF attack is included in the current version of IDIT-SDN. It can be found in */it-sdn/applications/attack-fdff.c*.

Table 1. Simulation Parameters

Simulation parameters	
Simulation time	36000 s
Data traffic rate	1 packet/ min [payload: 10 bytes]
Attack begins after	8400 s
***NSec* parameters**	
Monitoring window (`m_win`)	200
Detection window (`d_win`)	50
Sampling period	60 s
Critical value (`ca`)	2.82
sensitivity (`g`)	0
***CSec* parameters**	
Monitoring period	60 s
`S_SKIP`	10 samples
`T_WIN`	210 samples
`GAMMA`	0.45
`S_VAL`	0.95

Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Luz, T. C., Nunez, G. A., Margi, C. B., and Verdi, F. L. (2019). In-network performance measurements for software defined wireless sensor networks. In *2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*, pages 206–211.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Naous, J., Stutsman, R., Mazieres, D., McKeown, N., and Zeldovich, N. (2009). Delegating network security with more information. In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, page 19–26, New York, NY, USA. ACM.

Segura, G. A. N., Chorti, A., and Margi, C. B. (2022). Centralized and Distributed Intrusion Detection for Resource-Constrained Wireless SDN Networks. *IEEE Internet of Things Journal*, 9(10):7746–7758.

Segura, G. A. N., Margi, C. B., and Chorti, A. (2019). Understanding the Performance of Software Defined Wireless Sensor Networks Under Denial of Service Attack. *Open Journal of Internet Of Things (OJIOT)*. Special Issue: Proc. Int. Workshop Very Large Internet of Things (VLIoT 2019) in conjunction with the VLDB 2019.

Segura, G. N. (2021). *Cooperative Intrusion Detection for Software-Defined Resource-Constrained Networks*. PhD thesis, Universidade de Sao Paulo.

Skaperas, S., Mamatas, L., and Chorti, A. (2019). Real-time video content popularity detection based on mean change point analysis. *IEEE Access*, 7:142246–142260.