Policy and Network-based Intrusion Detection System for IPv6-enabled Wireless Sensor Networks

João P. Amaral[1], Luís M. Oliveira[1,2], Joel J. P. C. Rodrigues[1,3], Guangjie Han[4], Lei Shu[5]
[1] Instituto de Telecomunicações, University of Beira Interior, Portugal
[2] Polytechnic Institute of Tomar, Portugal
[3] University ITMO, St. Petersburg, Russia
[4] Hohai University, Changzhou, People's Republic of China
[5] Guangdong University of Petrochemical Technology, Maoming, People's Republic of China

{jpgamaral; loliveira}@it.ubi.pt; joeljr@ieee.org; hanguangjie@gmail.com; lei.shu@lab.gdupt.edu.cn

Abstract— **The recent years realize a progressive transition where fixed computing reached maturity and the mobility age started to thrive. Nowadays, another transition from the mobility age to the "Internet of Everything" (IoE) is taking place. In the IoE vision, several types of quotidian objects will be able to communicate over the Internet. As a result, it is expected that within a decade, IoE will have an economic value of $14.4 trillion, as the number of devices connected to the Internet continues to increase exponentially. The support for security services in these emerging resource-constrained devices is considered a challenge but needs to take into account from the very early stages of the wireless network inception. This paper proposes a network-based intrusion detection system (IDS) for IPv6-enabled wireless sensor networks. The proposed IDS is used to detect security attacks based on traffic signatures and abnormal behaviors.**

Keywords— **6LoWPAN; Intrusion detection system; IDS; Internet of things; Internet of everything; Wireless sensor networks; WSN**

I. Introduction

With recent advances in micro-electromechanical systems technology, wireless communications, and digital electronics, it is technically and economically available manufacture small and low cost sensor devices in large scale. These devices are characterized by small size, small computing and storage resources, power constrains, reduced radio ranges and data throughput. Several types of transducers can be connected to these devices, turning them suitable to be used on monitoring and physical phenomena control in large-scale environments [1]. These characteristics can be used in a wide range of applications, including military, environmental monitoring, health, and home automation. A single network may comprise hundreds of sensor devices working together to accomplish a common task. The self-organization, fault-tolerance and self-optimization are the main characteristics of these networks [1]. Nowadays, there are many technologies that can be used to connect sensor devices, most of them based on IEEE 802.15.4 layer two protocol [2, 3]. Connecting these devices to the Internet is considered simultaneously an opportunity and an important challenge. An opportunity because the services provided by these networks can be globally accessed from any device with Internet connectivity; and a challenge because these networks are exposed to a wide range of potential security attacks. In fact, sensor networks are more prone to security attacks than regular networks. First, due to the resource constrains, these devices cannot support the computational overhead necessary to run most of the typical defensive mechanisms. Second, a single network can scale to hundreds of sensor nodes without any fixed network infrastructure and are commonly installed in harsh and unattended environments [3, 4]. Besides the differences, confidentiality, availability, integrity, authentication, non-repudiation, authorization, and data freshness are the most important security requirements for this type of networks. Then, these security requirements are be identified in the following security aspects:

- **Confidentiality**: to make information only accessible to the legitimate users [5]. The data secrecy must be ensured while it is stored on devices, when it is transmitted and also when it reaches the destination [1].
- **Availability**: to ensure reliability, survivability, and timely access of security objectives (data, resources, and network services) to legitimate users when needed despite denial-of-service attacks [3, 6].
- **Integrity**: to certify that received data was not changed in transit by third parties [5, 7] and to provide the information and sensor network systems the assurance of the accuracy and reliability.
- **Authentication**: to enable a node to verify the identity of the other nodes [5, 7].
- **Non-repudiation**: to ensure that a node cannot deny in the future the authorship of a message previously sent [1].
- **Authorization**: to ensure that only authorized nodes can have access to network services or resources [7].
- **Freshness**: to prevent old messages from being replayed without being detected [1].

Since wireless sensor node replacements is a common situation, it is also important to guarantee the following [1]:

- **Forward secrecy**, a sensor node should not be allowed to have information related to future messages after effectively leaving the network.
- **Backward secrecy**, a new joined sensor should not be able to know any previously transmitted message.

The WSN vulnerabilities are such as a serious problem that some experiments concluded that an intruder could effectively interpose itself in the network within five minutes [8, 9]. These experiments also concluded that the pivot point in nearly all the

978-1-4799-2003-7/14/$31.00 ©2014 IEEE

attacks are exploited weaknesses found in routing protocols of WSNs. Usually, the primary task for an attacker against sensor networks is to establish itself as a legitimate node within the network [2]. It is possible to increase the security in WSNs using authentication, cryptography or key management in order to protect the sensor network against a number of different attacks from external nodes, but these security solutions do not protect against malicious internal nodes, which already have the required cryptographic keys [10]. In fact, these security solutions alone cannot prevent all the possible attacks and fulfill every security service. Since many attacks, in WSNs, are instigated by harmful nodes [11,12], an intrusion detection system (IDS) can provide protection from both internal and external attackers [2, 10, 13].

The remainder of this paper is organized as follows. Section 2 addresses the intrusion detection mechanisms state-of-the art, while section 3 proposes an architecture for a WSN network-based intrusion detection system. Section 4 focuses on the system evaluation and demonstration. Finally, section 5 concludes the paper and identifies future research topics.

II. Intrusion Detection Mechanisms

An intrusion detection system (IDS) is defined as a system that tries to detect and alert the occurrence of potential intrusions into a system or a network. The intrusions identification are based on node and network monitoring [9]. It can also be used to detect misbehaving nodes and to inform neighbor nodes to take proper countermeasures. The current detection mechanisms are implemented in specific elements known as IDS agents [10, 14]. The intrusion detection systems can be classified as intrusion detection systems and as intrusion protection systems, based on the reaction when an intrusion is detected. The intrusion detection systems only reports the abnormal activity detected. The intrusion protection systems also detect abnormal activities and reacts in order to stop the attack.

Conventionally, intrusion detection systems were classified into two classes: network-based (NIDS) and host-based IDS (HIDS) [15]. The former operate by overhearing the network's activity, capturing and investigating individual packets in order to raise alarms. NIDSs often require dedicated hosts and special equipment. The latter bases its operation on the specific activity experienced by each individual host. Host-based security solutions are able to detect continuous failed access attempts or modifications in resources that are vital to the normal operation of the system [15]. The main drawback of the HIDS method is the necessity of installation of the security solution on every host that belongs to the network. Moreover, each HIDS instance should be tailored to the specific host configurations [15].

An IDS can be classified as SIDS (Static IDS) or DIDS (Dynamic IDS). In the former, the attackers know which nodes run the IDS, which turns it easier to attack the network. In the latter, the nodes running the IDS alternate throughout the network lifetime [15]. Depending on the detection technique used, an intrusion detection system can be classified considering three main methodologies, described below.

A. *Signature detection*

Signature detection (also called misuse detection or rule-based detection) identifies an unauthorized use from signatures and consists in comparing audit data (e.g., action or behavior of nodes) with well-known attack patterns [10]. This technique monitors for the manifestation of a set of well-known signatures that indicate an intrusion and it is the predominantly technique used in classical IDS. However, it is not widely suitable for WSNs, because the numerous patterns that need to be defined in the system, as well as the computationally costly comparison algorithms, tend to hinder the node's longevity [16]. Misuse detection may exhibit low false positive rate and identify most known attacks in a rule database, but suffers from two noteworthy drawbacks: *i*) it requires knowledge to build attack patterns, and *ii*) it lacks flexibility by not performing well at discovering novel and previously unknown attacks [10]. This type of security solutions compares networks and hosts' activities against a set of known attack patterns, playing a very important role in networks with highly dynamic events, like WSNs. However, a frequently updated (and large) group of known attack patterns must be maintained, usually by the network's administrator, significantly jeopardizing the efficiency of this approach [10].

B. *Anomaly detection*

Anomaly detection systems identify an unauthorized use from analysis of an event. They first describe the current features of normal behavior of the network by detecting any activity that differs significantly from it. Afterwards, these systems mark any activity that deviate from the so-called 'normal' behaviors as intrusions. If a sensor node does not act accordingly to the defined specification of a particular protocol, the IDS would have high confidence to decide that the node is malicious [10]. Consequently, anomaly detection systems detect novel attacks more efficiently than misuse detection systems, but exhibit high rates of false positives for highly dynamic systems, like WSNs. The main ideas behind this type of security methodology are borrowed from statistical behavior modeling, which detects intrusions in a very accurate and consistent way, while presenting a low false positives rate, with the assumption that monitored system complies with some static behavioral patterns. However, it is not the case with WSNs. The normal behavior is usually established via automated training, a very expensive procedure for resource-constrained devices. The main pitfall of the anomaly detection strategy comes from the fact the system can exhibit legitimate but unprecedented behavioral activities, ultimately leading to a substantial false positives rate. On the other hand, a potential intrusion that does not exhibit anomalous behavior may pass unnoticed, increasing the false negatives number [10].

C. *Specification-based detection*

Specification-based detection is quite similar to anomaly detection, but the correct behavior of the network is manually defined, resulting in a smaller false positives rate. This methodology tries to extract the best of the previous two techniques, by trying to unfold deviations from normal behavioral patterns that are defined neither by machine learning techniques nor by training data [10]. However, the development of detailed specifications is difficult and makes it less flexible to the different environments [16]. The main drawback of this approach is the fact that the development of attack or protocol specifications is done manually, which is a time-consuming process for human beings. The specifications that describe a correct operation must be manually defined by the network administrator. This strategy exhibits yet another drawback, which is unfeasible to detect malicious behaviors that do not interfere with the set of defined specifications. In some particular cases, signature and anomaly-based detection techniques can be used alongside in the same IDS agent, originating hybrid detection solutions [10]. All the three aforementioned intrusion detection approaches can be used on host-based and network-based IDS systems.

The effectiveness of IDS solutions that were successfully designed and deployed for fixed wired networks are limited for wireless ad-hoc networks. Then, they cannot be applied directly to WSNs taking into account their specific network characteristics. In addition to the distinct characteristics that make the design of a security model for wireless sensor networks contrasting with other types of networks, WSNs also inherit all the aspects of wireless networks. The batteries may not be rechargeable or the network may be deployed in an unpredictable environment, thus, making it impossible to recharge or replace the sensor nodes' energy power sources [17].

An intrusion detection system is a paramount security mechanism against both internal and external intruders [18]. Particularly, in a resource-constrained type of network like WSNs, such a system should maintain an energy-efficient detection of untrustworthy or malicious nodes as its pivotal concern by isolating them from the network [10].

III. SYSTEM ARCHITECTURE OF THE PROPOSED IDS

The proposed intrusion detection system follows a network-based approach, which is the most suitable approach for networks with dynamic topologies and link constraints like wireless sensor networks. Selected network nodes acting as *watchdogs* are deployed with the network-based intrusion detection system (NIDS), with the purpose to identify the possible intrusions by eavesdropping the exchanged packets in their neighborhood. These nodes perform local packet monitoring, and their main task is to eavesdrop the exchanged messages between the nodes in their neighborhood, which act as host-based IDS. The monitored messages are matched against the set of rules configured in each NIDS agent. If a particular message matches a rule, an alarm is generated and sent to the EMS (Event Management System), as illustrated in Figure 1.

A wireless sensor network is a heterogeneous type of network and each NIDS is installed in a particular location of the network. So, all of them cannot be configured with the exact same set of rules. Therefore, each watchdog is tailored with a specific set of rules that should match as closely as possible its neighborhood traffic patterns. For example, for an edge router, it may be entirely reasonable for it to experience high rates of ICMPv6 messages (RPL, for routing, and NDP, for neighbor discovery and stateless address auto-configuration purposes) exchanged, but that might not be the case for an ordinary sensor node with limited radio reachability and few neighbors. In order to dynamically configure each NIDS with a particular set of rules, a policy programming approach was adopted. Using the Finger2 policy-programming engine built for TinyOS [19], it is possible to design policies that provide a flexible means of specifying adaptation strategy in pervasive systems and sensor networks. Typical policies include *authorization policies* which specify the conditions under which resources or services can be accessed by other devices; and *obligation policies* in the form of event-condition-action rules, which can be used to define the adaptive behavior to be executed when a failure occurs or context changes, what events or notification to generate to external entities, or action to be performed if a threshold is exceeded. These policies are essentially reactive rule-based systems for applying strategies in accordance to specific events in the system that are interpreted rather than hard-coded into software components. Therefore, they can be dynamically modified while the system is running without the encumbrance of reprogramming the sensor node, which presents an attainment considering resource-constrained monolithic operating systems for sensor networks, like TinyOS, the prevalent operating system for sensor nodes. Dynamic management of policies, i.e. loading, enabling, disabling, or removal of policies at runtime, enables dynamic strategy adaptation. The *Finger2* middleware runs on individual sensor nodes, thus allowing a constrained form of dynamic distributed reprogrammability [19].

Each rule belongs to a pre-defined *role group* and is created using an application specially developed for that purpose. A *role group* is a necessity in order to categorize the different traffic transmitted in the wireless sensor network. Each rule is

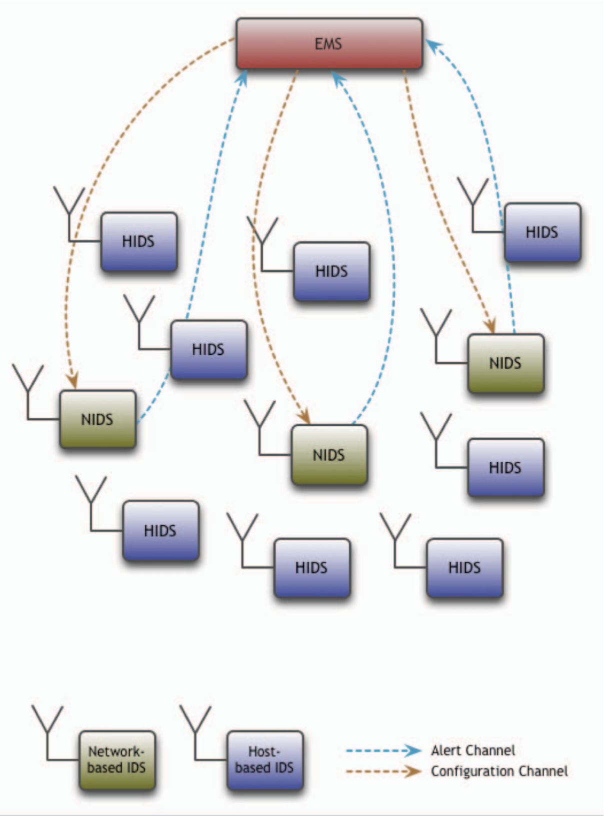


Figure 1: Illustration of the proposed system architecture.

transmitted to the watchdogs (NIDS nodes) using a configuration channel, represented by an UDP socket established in each node. In each NIDS there's a UDP shell running on port 2000 that allows us to send commands to the nodes containing the rules specifications. If a monitored message matches one of these rules, an alarm is raised and sent to the EMS, and when the number of alarms raised about a specific sensor node exceeds some pre-defined threshold, the sensor node should be isolated from the rest of the network and classified as a compromised node.

The system architecture is composed by two main application components: the network-based IDS application and an EMS application running on a workstation, as illustrated in Figure 1. The packet monitoring application used in our work comes bundled with the latest version of the TinyOS operating system and uses BLIP (Berkeley Low-power Internet Protocol) stack version 2, which is a novel and improved stack that supports the most recent improvements related to the 6LoWPAN ecosystem.

The EMS application runs on a node (possibly a *sink* node) with no resource or energy constraints and with full processing and storage capabilities. Its main purpose is to collect the events received from the NIDS nodes and correlate them in order to make decisions regarding possible compromised inside nodes or external intruders. Using the EMS graphical user interface, it is possible to have access to events reports and manage the nodes that run the NIDS application. Alongside the EMS application, another complementary application is run on the same machine that enables the network administrator to view the eavesdropped packets by the *sink* node (Figures 2 and 3).

A. Packet Monitoring Module

This module collects audit data, mainly communication activities within the radio range of each NIDS, which is provided to the detection module, by listening promiscuously the transmissions in the node's neighborhood.

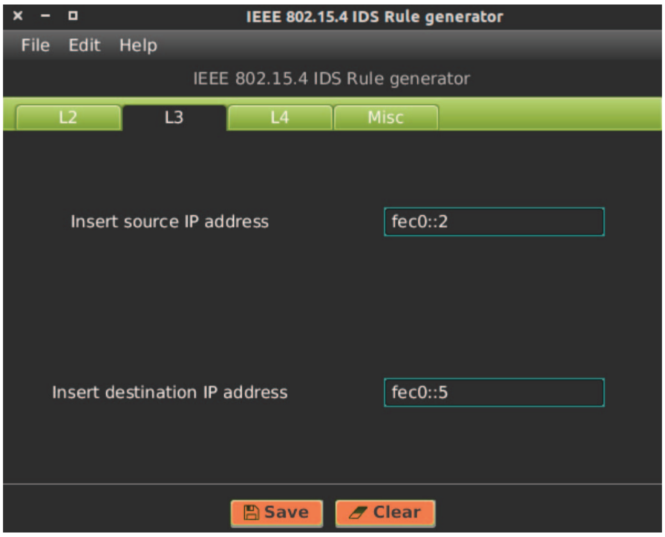
To reduce the overload of resources caused by running the NIDS application in each *watchdog*, there is no need for the watchdogs to store the eavesdropped packets or any other information in their memory. Basically, they temporarily buffer each packet in order to apply the rules defined by the local detection engine and see if any of these rules are satisfied. Afterwards, the packet can be discarded and no historical or statistical data need to be kept in the node's memory.

B. Detection Module

This module is responsible for storing, managing and applying all the rules that should trigger an intrusion alert. These rules are specified by the network administrator and are periodically sent to the watchdogs in order to improve the intrusion detection performance. The performance can be improved by analyzing the network traffic at different locations, avoiding the overhead caused by training the network regarding its normal behavior should be.

C. Action Module

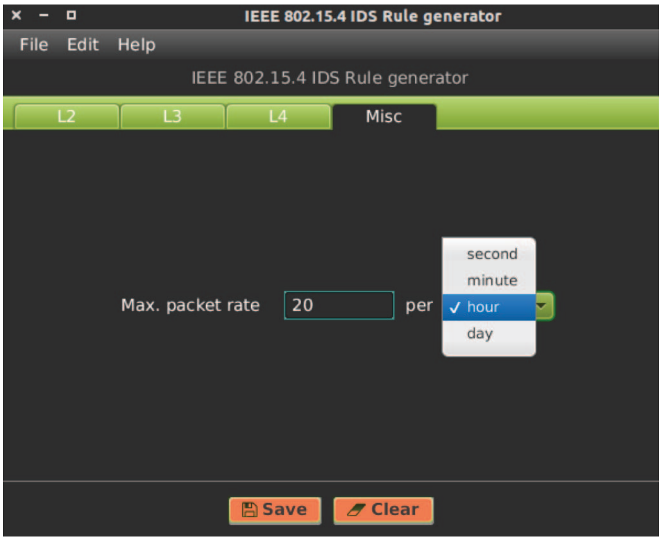
After a watchdog suspects that an intrusion is undergoing in its neighborhood, the action module sends an alert to the EMS. Afterwards, the network administrator can correlate this alert with other possible alerts sent by other watchdogs regarding the same specific node and take proper actions, e.g., completely restrain the compromised node from communicating with the rest of the network. The sensor network must remain resilient and fault-tolerant after intruders try to compromise it, and as such, changes in the routing paths and generation of new cryptographic keys should be carried away.


IEEE 802.15.4 IDS Rule generator
File
Edit
Help
IEEE 802.15.4 IDS Rule generator
L2
L3
L4
Misc
Save
Clear

Insert source IP address: fec0::2

Insert destination IP address: fec0::5

Figure 2: Rule generation (L3 fields).


IEEE 802.15.4 IDS Rule generator
File
Edit
Help
IEEE 802.15.4 IDS Rule generator
L2
L3
L4
Misc
second
minute
hour
day
Save
Clear

Max. packet rate: 20 per ✓ hour

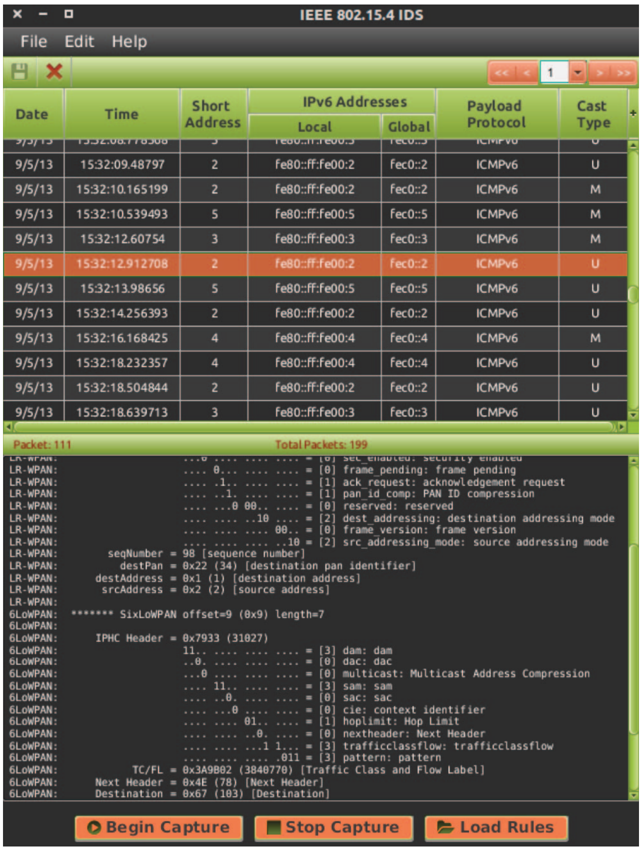
Figure 3: Rule generation (packet rate).

IV. System Evaluation and Demonstration

The system was deployed using TinyOS 2.1.2 and InteliJ IDEA 12.3.1 installed on an Ubuntu 13.04 "Raring Ringtail" virtual machine. In order to perform our experimentation, five Crossbow TelosB motes were used. A local network interface on the desktop side was used to route packets through a wireless sensor mote acting as base station (*sink* node) that had been programmed as an edge router using the PPPRouter application. This router, also running an UDP shell, provided the interface between IPv6 traffic on Ethernet networks, and the 6LoWPAN wireless sensor networks. Some packets eavesdropped by the *sink* node are displayed in Figure 4.

The NIDS application runs two component applications, both modified versions of the PPPSniffer application and the *Finger2IPv6* application, bundled with the latest TinyOS distribution. The former is a novel nesC application created at University of Bremen for TinyOS 2.1+ and BLIP 2 that receives and captures frames on a previously specified IEEE 802.15.4 channel and forwards them to a workstation using the Point-to-Point Protocol (PPP). The application was submitted to TinyOS repositories on October 25th, 2012 and was included in TinyOS main distribution on December 17th, 2012. It operates on an alternative radio stack for the TI CC2420 radio, using the CC2420X *rfxlink* library, which fixes a time stamping issue in the original CC2420 radio stack. This stack is IEEE 802.15.4 compliant and is able to operate with microsecond-precision time stamping, as well as 32khz time stamping, but does not support hardware acknowledgements or security. The latter one is a wrapper around the Finger2 middleware called *Finger2IPv6* and was used so that the commands that were invoked within of *Finger2* were architected to be invoked via the UDP shell provisions that come with the BLIP network stack.

Figure 5 illustrates an example of a rule generated with the developed application showcased in Figures 2 and 3. This rule was transmitted to the watchdogs deployed in the testbed. Each rule is saved locally in JSON (JavaScript Object Notation) format and sent to the watchdogs partitioned by the different fields in sequential UDP commands, due to the fact that the sensor nodes are incapable of efficiently parsing JSON. This approach lessens the computational burden of the sensor nodes, because it is no longer necessary to perform any rule parsing otherwise necessary if the rule was sent in JSON format.

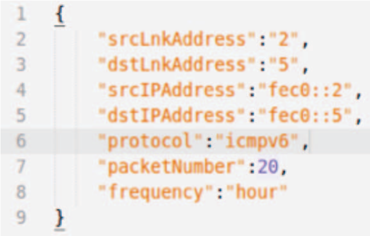


Date	Time	Short Address	IPv6 Addresses		Payload Protocol	Cast Type
			Local	Global		
9/5/13	15:32:08.778308	3	fe80::ff:fe00:3	fec0::3	ICMPv6	U
9/5/13	15:32:09.48797	2	fe80::ff:fe00:2	fec0::2	ICMPv6	U
9/5/13	15:32:10.165199	2	fe80::ff:fe00:2	fec0::2	ICMPv6	M
9/5/13	15:32:10.539493	5	fe80::ff:fe00:5	fec0::5	ICMPv6	M
9/5/13	15:32:12.60754	3	fe80::ff:fe00:3	fec0::3	ICMPv6	M
9/5/13	15:32:12.912708	2	fe80::ff:fe00:2	fec0::2	ICMPv6	U
9/5/13	15:32:13.98656	5	fe80::ff:fe00:5	fec0::5	ICMPv6	U
9/5/13	15:32:14.256393	2	fe80::ff:fe00:2	fec0::2	ICMPv6	U
9/5/13	15:32:16.168425	4	fe80::ff:fe00:4	fec0::4	ICMPv6	M
9/5/13	15:32:18.232357	4	fe80::ff:fe00:4	fec0::4	ICMPv6	U
9/5/13	15:32:18.504844	2	fe80::ff:fe00:2	fec0::2	ICMPv6	U
9/5/13	15:32:18.639713	3	fe80::ff:fe00:3	fec0::3	ICMPv6	U

Packet: 111 Total Packets: 199

```
LR-WPAN:                     ...0 .... .... .... = [0] sec_enabled: security enabled
LR-WPAN:                     .... 0... .... .... = [0] frame_pending: frame pending
LR-WPAN:                     .... .1.. .... .... = [1] ack_request: acknowledgement request
LR-WPAN:                     .... ..1. .... .... = [1] pan_id_comp: PAN ID compression
LR-WPAN:                     .... ...0 00.. .... = [0] reserved: reserved
LR-WPAN:                     .... .... ..10 .... = [2] dest_addressing: destination addressing mode
LR-WPAN:                     .... .... .... 00.. = [0] frame_version: frame version
LR-WPAN:                     .... .... .... ..10 = [2] src_addressing_mode: source addressing mode
LR-WPAN:         seqNumber = 98 [sequence number]
LR-WPAN:           destPan = 0x22 (34) [destination pan identifier]
LR-WPAN:       destAddress = 0x1 (1) [destination address]
LR-WPAN:        srcAddress = 0x2 (2) [source address]
LR-WPAN:
6LoWPAN:  ******* SixLoWPAN offset=9 (0x9) length=7
6LoWPAN:
6LoWPAN:       IPHC Header = 0x7933 (31027)
6LoWPAN:                     11.. .... .... .... = [3] dam: dam
6LoWPAN:                     ..0. .... .... .... = [0] dac: dac
6LoWPAN:                     ...0 .... .... .... = [0] multicast: Multicast Address Compression
6LoWPAN:                     .... 11.. .... .... = [3] sam: sam
6LoWPAN:                     .... ..0. .... .... = [0] sac: sac
6LoWPAN:                     .... ...0 .... .... = [0] cie: context identifier
6LoWPAN:                     .... .... 01.. .... = [1] hoplimit: Hop Limit
6LoWPAN:                     .... .... ..0. .... = [0] nextheader: Next Header
6LoWPAN:                     .... .... ...1 1... = [3] trafficclassflow: trafficclassflow
6LoWPAN:                     .... .... .... .011 = [3] pattern: pattern
6LoWPAN:             TC/FL = 0x3A9B02 (3840770) [Traffic Class and Flow Label]
6LoWPAN:       Next Header = 0x4E (78) [Next Header]
6LoWPAN:       Destination = 0x67 (103) [Destination]
```

Figure 4: Eavesdropped Packets Information.



```
{
    "srcLnkAddress":"2",
    "dstLnkAddress":"5",
    "srcIPAddress":"fec0::2",
    "dstIPAddress":"fec0::5",
    "protocol":"icmpv6",
    "packetNumber":20,
    "frequency":"hour"
}
```

Figure 5: Example of a rule.

The creators of the *Finger2* middleware state that the total memory requirements are 12.23 KB in ROM and 0.72 KB of RAM. Particularly for the RAM requirements, the number does not include storage requirements of policies, as they are application specific and should not be included as part of the middleware core. The minimum memory footprint of a policy is 24 bytes but there is not an upper size, it may include an arbitrary number of actions. However, the majority of policies are not expected to exceed about 50 bytes and it is typically expect, at most, a few tens of policies in a mote. The current generation ranges between 48-128 KB of ROM and 4-16 KB of RAM, which leaves enough memory space for developers to build applications on top of *Finger2* [19].

V. CONCLUSION AND FUTURE WORK

Wireless sensor networks are rapidly becoming a technology with high potential to be applied in a plethora of scenarios, such as home-automation, industrial, and medical/healthcare environments. Both WSN networks and resource-unconstrained networks share almost the same security requirements. However, the nodes resource constraints, the number of nodes and the absence of an organized communication infrastructure, makes the support of security services in WSN networks more challenging when compared with resource unconstrained networks. This paper presented an intrusion detection system that can be used as a complement to other available security mechanisms to detect and to report security attacks. Instead of limiting this proposed system to a pre-defined and specific type of attack, it tries to model possible traits of incorrect behavior in a particular region of a WSN.

This paper proposed a network-based approach in order to perform intrusion detection in IPv6-enabled wireless sensor networks. It takes the commodities advantage of a well-known protocol suite like the IP stack (e.g., UDP commands). Then, it was possible to create a standards-compliant system that can be deployed in a variety of different WSNs, so long as they support the IP stack to a certain degree. Moreover, a *testbed* was created to validate the proposed intrusion detection system. In this proposal, two TinyOS applications (*PPPSniffer* and *Finger2IPv6*) were modified in order to capture the eavesdropped packets by a node and compare them against a set of rules configured in a particular node. By using a policy programming approach, it was possible to modify and adjust a particular NIDS agent installed in a node reacting to changes in its neighborhood traffic exchange patterns without requiring over-the-air programming and complete re-installation of the node's image.

The carried work still requires improvements of the proposed mechanism. First, it is necessary to add confidentiality, authenticity, integrity, and data freshness guarantees to protect the configuration and the event channel. Second, a new firmware may be developed for being installed in the IDS sensor nodes to dynamically detect the wireless channel used by the WSN nodes to communicate, because in the current version these tasks are performed manually. Finally, it is necessary to optimize the process used to enforce and to store new detection rules. These are suggestions for further work.

Acknowledgments

This work has been partially supported by *Instituto de Telecomunicações*, Next Generation Networks and Applications Group (NetGNA), Portugal, by National Funding from the FCT - *Fundação para a Ciência e a Tecnologia* through the PEst-OE/EEI/LA0008/2013 Project, by Government of Russian Federation, Grant 074-U01, and by the AAL4ALL (Ambient Assisted Living for All), project co-funded by COMPETE under FEDER via QREN Programme.

References

[1] X. Chen, K. Makki, K. Yen, N. Pissinou, “Sensor network security: a survey”, IEEE Communications Surveys and Tutorials, IEEE, vol. 11, no. 2, pp. 52 – 73, Second Quarter 2009.

[2] I. Onat, A. Miri, “An Intrusion Detection System for Wireless Sensor Networks”, IEEE International Conference on Wireless and Mobile Computing, Networking And Communications (WiMOB 2005), Montreal, Canada, August 22-25, vol. 3, pp. 253-259, 2005, DOI: 10.1109/WIMOB.2005.1512911.

[3] Y. Zhou, Y. Fang, Y. Zhang, “Securing Wireless Sensor Networks: A Survey”, IEEE Communications Surveys and Tutorials, vol. 10, no. 3, pp. 6-28, 2008.

[4] A.-S. K. Pathan, H.-W. Lee, C.S. Hong, “Security in Wireless Sensor Networks: Issues and Challenges”, in 8th International Conference on Advanced Communication Technology (IEEE ICACT 2006), Volume II, 20-22 February, Phoenix Park, Korea, 2006, pp. 1043-1048.

[5] W. Stallings. “Cryptography and Network Security - Principles and Practices”, 3rd ed. Prentice Hall, Upper Saddle River, NJ, 2003.

[6] L. Oliveira, J. Rodrigues, A. Sousa, J. Lloret. “A Network Access Control Framework for 6LoWPAN Networks”, Sensors, MDPI, vol. 13, no. 1, pp. 1210-1230, 2013.

[7] Y. Wang, W. Lin, T. Zhang, “Study on security of wireless sensor networks in smart grid”, International Conference in Power System Technology (POWERCON 2010), Hangzhou , China, October 24-28, 2010, pp. 1–7, 2010.

[8] I. Krontiris, T. Dimitriou, F.C. Freiling, “Towards Intrusion Detection in Wireless Sensor Networks”, in 13th European Wireless Conference, Paris, France, April 1-4, 2007.

[9] G. Huo, X. Wang, “DIDS: A Dynamic Model of Intrusion Detection System in Wireless Sensor Networks”, International Conference on Information and Automation (ICIA 2008), Changsha, June 23-28, 2008, pp. 374-378, DOI: 10.1109/ICINFA.2008.4608028.

[10] A. Abduvaliyev, A.-S.K Pathan, J. Zhou, R. Roman, W. Wong, “On the Vital Areas of Intrusion Detection Systems in Wireless Sensor Networks,” IEEE Communications Surveys & Tutorials, vol.15, no.3, pp.1223-1237, Third Quarter 2013, DOI: 10.1109/SURV.2012.121912.00006.

[11] A. Agah, S.K. Das, K. Basu, M. Asadi, “Intrusion Detection in Sensor Networks: a Non-Cooperative Game Approach”, in 3rd IEEE International Symposium on Network Computing and Applications (NCA 2004), August 30 - September 1, 2004, pp. 343-346, DOI: 10.1109/NCA.2004.1347798.

[12] I. Krontiris, T. Dimitriou, T. Giannetsos, M. Mpasoukos, “Intrusion Detection of Sinkhole Attacks in Wireless Sensor Networks”, LNCS, vol. 4837, pp. 150-161, 2008.

[13] H.Y. Lin, T.C. Chiang, “Intrusion Detection Mechanisms Based on Queuing Theory in Remote Distribution Sensor Networks”, Advanced Materials Research, vol. 121-122, June 2010, pp. 58-63, DOI: 10.4028/www.scientific.net/AMR.121-122.58.

[14] P. Techateerawat, A. Jennings, “Energy Efficiency of Intrusion Detection Systems in Wireless Sensor Networks”, IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops (WI-IAT 2006 Workshops), Hong Kong, 2006, pp. 227-230, DOI: 10.1109/WI-IATW.2006.65.

[15] O. Kachirski, R. Guha, D. Schwartz, S. Stoecklin, E. Yilmaz, “Case-based agents for packet-level intrusion detection in ad hoc networks”, 17th International Symposium on Computer and Information Sciences, CRC Press, pp. 315–320, October 2002.

[16] L. Besson, P. Leleu, “A distributed intrusion detection system for ad-hoc wireless sensor networks: The AWISSENET Distributed Intrusion Detection System”, 16th International Conference on Systems, Signals and Image Processing (IWSSIP 2009), June 18-20, 2009, pp. 1-3, DOI: 10.1109/IWSSIP.2009.5367767

[17] P. Brutch, C. Ko, “Challenges in intrusion detection for wireless ad-hoc networks,” 2003 Symposium on Applications and the Internet Workshops, January 27-31, 2003, pp. 368-373, DOI: 10.1109/SAINTW.2003.1210188.

[18] I. Krontiris, Z. Benenson, T. Giannetsos, F.C. Freiling, T. Dimitriou, “Cooperative Intrusion Detection in Wireless Sensor Networks”, in EWSN 2009, LNCS, vol. 5432, pp. 263-278, 2009.

[19] T. Bourdenas, M. Sloman, “Starfish: policy driven self-management in wireless sensor networks”, ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2010), ACM, New York, NY, USA, pp. 75-83, 2010, DOI: 10.1145/1808984.1808993.