to appear in M2MSec - Workshop on Security and Privacy in Machine-to-Machine Communications, in conjunction with IEEE Conference on Communications and Network Security (CNS) 2014, October 29-31, 2014, San Francisco, CA, USA

Integration and Evaluation of Intrusion Detection for CoAP in Smart City Applications

Jana Krimmling
IHP, Frankfurt (Oder), Germany
krimmling@ihp-microelectronics.com

Steffen Peter
University of California, Irvine, USA
st.peter@uci.edu

***Abstract*—The Constrained Application Protocol (CoAP) is a promising candidate for future smart city applications that run on resource-constrained devices. However, additional security means are mandatory to cope with the high security requirements of smart city applications. We present a framework to evaluate lightweight intrusion detection techniques for CoAP applications. This framework combines an OMNeT++ simulation with C/C++ application code that also runs on real hardware. As the result of our work, we used our framework to evaluate intrusion detection techniques for a smart public transport application that uses CoAP. Our first evaluations indicate that a hybrid IDS approach is a favorable choice for smart city applications.**

***Index Terms*—Smart city; Critical Infrastructure protection; Intrusion detection; Simulation; OMNeT++**

I. Introduction

Smart city applications such as smart buildings, or smart public transport can improve our everyday life and comfort. City administrations, businesses, and citizens will benefit from easier information sharing as well as optimized processes and control. Nevertheless, security and safety are essential requirements for smart city applications since they may operate on Critical Infrastructures.

For their task, smart city applications interact with various networks and devices. A smart city application may send data over public networks such as the internet, cellular networks, networks that belong to different operators, or self-organized wireless sensor networks. The devices used in these networks vary from large cloud and network servers, through customer equipment, down to small sensor nodes. This heterogeneous structure poses a variety of challenges, in particular applying security to such systems. Therefore, the protection of smart city applications is essential since they will form parts of Critical Infrastructures. Protection must preferably be made equally available on small sensor nodes as well as on powerful servers.

In this paper we apply the Constrained Application Protocol (CoAP) [1] as the underlying communication protocol for a smart city application. CoAP is a very lightweight, simple, and effective communication protocol that meets the needs of resource-constrained embedded devices. It can be considered as a simplified Hypertext Transfer Protocol (HTTP) service that works on top of the Internet Protocol (IP) and the User Datagram Protocol (UDP) and can be processed on all internet-ready devices.

The Constrained Application Protocol (CoAP) already supports a range of message security means. It can be protected using either Datagram Transport Layer Security (DTLS) or Internet Protocol Layer Security (IPsec). Both security protocols provide integrity and confidentially using encryption and authentication between the participating devices. However, a number of security weaknesses have already been discussed [2]. In addition, compromised devices or sensors are not protected by the embedded means in any case. Lightweight intrusion detection systems (IDSs) can complement available security means to identify potential security issues, warn administrators, or to act automatically to disable threats. While literature is rich in IDS approaches, a direct application of existing intrusion detection systems is at least questionable. The related IDS techniques are either not designed for resource-constrained devices, not suitable for the intended application scenarios, or have to be adapted to CoAP.

In this paper we investigate how intrusion detection can be applied to smart city applications that use the Constrained Application Protocol (CoAP). Our aim is not to develop new intrusion detection system (IDS) approaches but to provide a framework that allows developers to evaluate proposed IDS techniques for intended smart city applications. For that purpose, we introduce a simulation approach with integrated C/C++ application and IDS code that also runs on real hardware devices. Our approach uses the OMNeT++ network simulation environment [3]. With this simulation approach it is possible to combine small scale tests of smart city and IDS applications on real hardware devices with the complex modeling possibilities that only simulations can provide.

The contributions of this paper are the following:

- A software synthesis C/C++ framework to port a smart city application, a CoAP communication library, an IDS library, and an attacker library to both, hardware devices and an OMNeT++ simulation that allows us to perform systematic tests with specific attacker models (Section III).
- Integration of three IDS approaches for CoAP in Section IV. Applying our framework, the IDSs proved to perform correctly in the simulation and on real hardware devices.
- Systematic evaluation of the IDS approaches in Section V for a specific smart city scenario that is introduced next. The results indicate that a hybrid IDS approach is a favorable choice for smart city applications.

II. A Public Transport Scenario

In this section we introduce a smart public transport scenario in Subsection A as a motivating example and as a basis to apply our framework. The example uses the Constrained Application Protocol (CoAP), therefore we give a short overview on CoAP in Subsection B. Afterwards we discuss security related issues in smart city applications in Subsection C and state-of-the-art protection means in Subsection D.

A. Public Transport Example

In the smart public transport scenario positions and travel times of buses are collected in a cloud system and evaluated to improve business processes and to provide services to passengers and local authorities. Citizens can use the web services in the cloud using their mobile phones to buy tickets, plan routes, or get departure times and delays of buses on their travel route. Local authorities or the public transport coordinator can use the cloud services to react to bus delays faster, reschedule buses in a short time, or display the delay information for the public transport users.

Figure 1 shows the architecture of this scenario. Several mobile nodes (MN) with sensors (small circles) are placed on buses. These devices are connected to a 3G/4G cellular network via its base station controller (BSC). The mobile nodes send their sensor values, such as Global Positioning System (GPS) data, to a central control station (CCS). The BSC forwards those messages to the CCS that resides in another network. The sensor data is transferred to a cloud system (Cloud) in the internet to offer the web services to the citizens and the local authorities.

The mobile nodes talk to the central control station and the cloud system using the Constrained Application Protocol (CoAP), whereas the end users may access the data via standard Hypertext Transfer Protocol (HTTP) on their mobile phones. For the practical evaluation of our framework we focus on the CoAP part of this network. Our hardware setup contains Raspberry Pi development boards for the implementation of the mobile nodes and a Linux Notebook as the central control station.

B. Introduction to the Constrained Application Protocol

The Constrained Application Protocol (CoAP) is very lightweight, simple, and effective. It brings a simplified Hypertext Transfer Protocol (HTTP) similar service to resource-constrained embedded devices. In contrary to HTTP, CoAP uses the underlying User Datagram Protocol (UDP) instead of the Transmission Control Protocol (TCP). Therefore, CoAP is well suited for networks with loosely coupled (wireless) connections such as described in the example scenario above. Loosely coupled means that some messages can get lost, be duplicated, or modified by reflection, refraction, or diffraction in the environment. CoAP provides confirmable and non-confirmable messages and a stop-and-wait retransmission mechanism with exponential back-off time to protect against duplicate and lost messages.

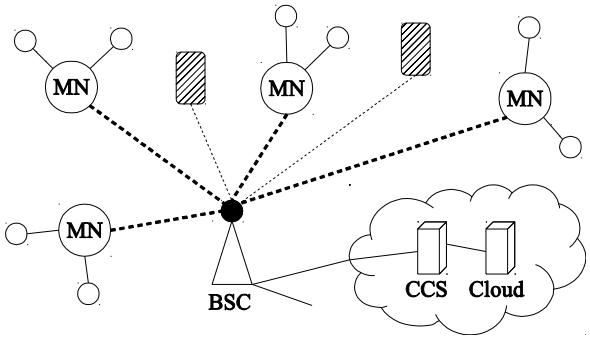

Figure 1: General architecture of the public transport example. Several mobile nodes (MN) placed on buses send their sensor data to a central control station (CCS) with connected cloud system that offers web services. Citizens access the cloud services via their mobile phones (grey rectangles).

C. Smart City Security Considerations

The introduced example scenario is challenged by a range of security related issues that can arise either from anomalies or from attacks. Anomalies are errors that typically occur naturally and are non-persistent such as faulty sensor values, device malfunctions, or communication errors. Anomalies can also result from attacks, in which case they are persistent. Attacks can be executed on sensors, devices, the communication channel of devices, or on a device's software by a malware infection.

Sensor **anomalies** occur if sensor measurements are temporarily faulty. In this case the faulty sensor value is delivered to the cloud where it may produce errors in the smart city application. Device malfunctions can result from bit flips in a device's memory that were caused by natural radiation. In this case a faulty value is processed and can lead to software errors and therefore to device malfunctions. Communication errors are frequent in loosely coupled (wireless) networks since the environment will add noise and interferences caused by reflection, refraction, or diffraction to the communication channel through which a message is transmitted. This can lead to lost, duplicated, or unintentionally modified messages.

Physical **attacks** on sensors and devices can be performed by destruction, using force, influencing the environment, or side-channel attacks. Whereas destruction and using force produce noticeable results, influence on the environment and side-channel attacks may be hidden and left undiscovered. Cyber-attacks on the communication channel of devices can be performed in numerous ways by espionage and message manipulation. Attackers can also perform physical attacks on a communication channel by producing destructive interferences with it. A malware infection can disturb or destroy a device's software or firmware and will result in unknown device behavior. Malware infections can affect the whole network operation and lead to serious errors in the smart city application.

Each issue discussed in this section may significantly compromise the service quality of the smart city application, which calls for appropriate protection and defense mechanisms.

D. State-of-the-Art of Network Protection

To protect the smart city application from the variety of attacks discussed in the previous subsection, security means such as encryption, authentication, and authorization are supported using IPsec [4] or DTLS [5] mentioned in Section I. However, these measures do not consider the threat of compromised devices or sensors. Intrusion detection systems (IDS) [6][7][8] can not only help to monitor and detect security related issues but they will also be the only security measure left if other measures are bypassed, broken, or not available at all. While an IDS intrinsically does not actively protect the system, it helps to detect attacks that otherwise may left undiscovered for a longer period of time and as well form an underlying system for filtering [11] and service protection.

There is a variety of frameworks for different evaluation and simulation problems in smart sensor networks but only few frameworks examine IDSs for resource-constrained devices on real hardware devices. Even less frameworks use CoAP as the communication protocol for their networks [9]. Kasinathan [10] specifically examined detection possibilities for Denial-of-Service (DoS) attacks in 6LoWPAN networks and implemented an IDS framework [11] for that. There is a test framework by Raza [12] that uses real hardware devices and that is specifically targeted to intrusion detection in CoAP. Raza implemented an IDS framework for 6LoWPAN on the ContikiOS that primarily targets routing attacks. However, the mentioned approaches assume that they have access to the border router of the network to place heavyweight IDS parts there. This is not possible in our smart city example since the messages are routed over a cellular station that belongs to the network of another owner.

III. Synthesis and Test of CoAP Applications

As motivated in Section I, lightweight intrusion and anomaly detection systems (IDS) are needed to complement available security measures for a specific smart city application. Our aim is to test a variety of applications, attacker scenarios, networks and IDS configurations. In this section we present a software synthesis framework that generates application-specific C/C++ code, which includes code for CoAP and IDSs. The C/C++ code can then be compiled for the simulation environment or directly for the target hardware platform.

A. Framework

The flow of our synthesis framework is shown in Figure 2. The application is split up in modules that can be used in the simulation and on the real target platform. That way only the application scenario itself (the network topology) has to be manually adapted in the simulation.

The primary inputs of the flow consist of the application code with the application logic, the CoAP library, which implements the CoAP communication protocol, and one IDS that is selected from the IDS library, which provides different intrusion detection mechanisms.

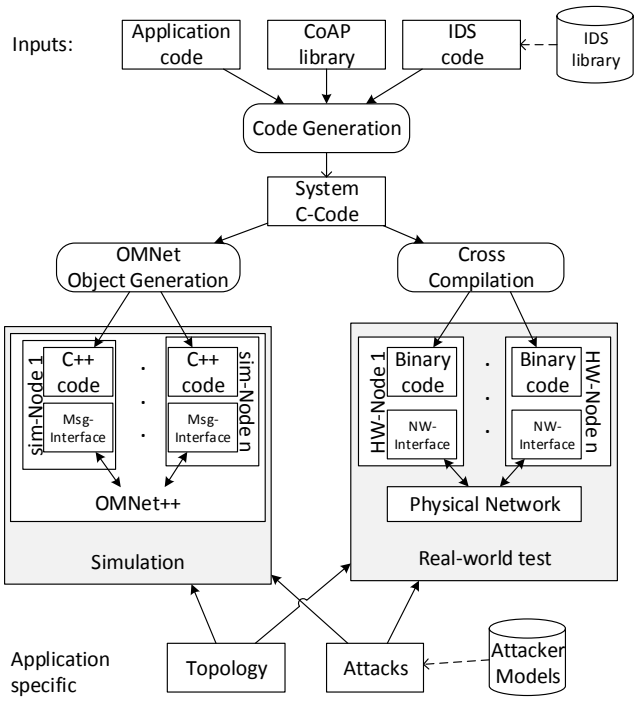

Figure 2: Framework design flow - the C/C++ code for the system is generated from the application code, the CoAP library and the IDS. This code can be executed as part of the OMNeT++ simulation or be compiled for the target hardware platform. Application-specific scenarios and attacks may be applied.

The inputs can be compiled as part of the code generation step to C/C++ code. This generic C/C++ code can be directly cross-compiled for our real target platform. The resulting binary runs on the hardware devices that communicate using the network interfaces of the devices. Alternatively, the C/C++ code can be translated into OMNeT++ object code. Technically, the code will be wrapped in a standardized C++ class which is understood by the OMNeT++ simulation environment. OMNeT++ is a free simulation environment for (wireless) network applications that provides a graphical user interface (GUI) for the visualization of the application scenario. In our framework, OMNeT++ implements the network and message forwarding between the simulated nodes.

The advantage of the simulation is the possibility to test a variety of attacks for application-specific topologies and scenarios. For a given application scenario and a selected IDS, the simulation injects errors, which are expected to be detected by the IDS. The detection success rate can be obtained from log files after the simulation. The attacks are selected from a library of reusable attacker models. The attack library is discussed in Subsection C.

The result of the process is a functional validation of the application and a security assessment of the application and the used IDS.

B. Hardware Abstraction

Since the C/C++ code is applied to real hardware devices and in the OMNeT++ environment, we need at least a hardware abstraction layer (HAL) for the message communication. We implemented such a HAL for the virtual messaging system in the OMNeT++ environment and as a set of drivers for the network interfaces on the real nodes. Both implementations provide the messages as byte streams to the upper application levels. That way the application code, CoAP library, and IDS can use the HAL for message communication and are therefore independent of the underlying system.

C. Attacker Models

One core functionality of the framework is the execution and management of attacks and anomalies in both the simulation and the real world test environment. In the following we describe a set of attacks and discuss their integration in the simulation and for the real world tests.

We can classify three categories of cyber-attacks: anomalies that result from attacks, routing attacks, and security attacks. A **sensor anomaly** is basically a faulty measurement value that can arise either out of an abnormal condition at a sensor or at the processing device due to a natural phenomenon such as radiation. Sensor anomalies can also be the consequence of an attack and lead to malfunctions of the smart city application. **Routing attacks** are focused on a change of the routing behavior of a network, mostly to disturb its operation. This is done by replaying, inserting, deleting, or disturbing messages to force individual nodes to change their routing behavior. **Security attacks** try to bypass protocol checks and security measures to achieve a special goal such as steeling or manipulating data. Security attacks involve a combination of a routing attack and a modification of values in messages such as an insertion attack with a specially modified message to provoke a certain reaction of a node.

Simple **routing attacks** such as replaying, inserting, or dropping messages are easy to implement on real nodes and in the simulation. A simple routing attack can be easily performed by placing an attacker node in the network that randomly sends replayed messages, inserts messages, or drops messages in the network. More complex routing attacks such as a Sybil attack involve targeted replaying, insertion, or dropping of messages. Such behavior can be easier to model in a simulation environment since complex routing attacks in a real world test scenario have to be synchronized with the network's operation. Jamming attacks try to disturb nodes by interfering with the nodes communication using an attacker node. Either a whole area with nodes is attacked by continuously sending random noise (simple jamming attack) or targeted nodes are attacked by specifically interfering with the communication channel in their sending time (targeted jamming attack). Modeling jamming attacks in a simulation is in general difficult since a detailed model of a communication channel is needed or another framework such as INET [13] has to be integrated in the simulation, which models a communication channel. Of course, this makes the simulation more complex to handle. In addition, targeted jamming attacks are difficult to model in both test environments since this attack needs synchronization with the network's operation as well.

Security attacks are very specific to the attacked node, the used protocol, and to the application of the network. In most cases a security attack will be a targeted Man-in-the-Middle attack that can be complicated to execute. A Man-in-the-Middle Attack is performed if messages from one node A to another node B are modified by an attacker node without the notice of both A and B. The most feared simple security attacks are flooding and Denial-of-Service attacks since they are easy to perform and can seriously harm a network's operation. Both attacks involve a massive amount of messages that are send to the targeted nodes. This draws computing time for message processing on these nodes. When attacked, the targeted nodes or the network cannot fulfill its original purpose anymore because the devices try to process a huge amount of messages that were sent by the attacker. This behavior can be modelled in a real world test environment by placing an attacker node in the network that continuously sends a huge amount of messages to one or more nodes in the network. This attack is more difficult to model in a simulation environment since a lot of timing related issues have to be modeled for the nodes, the communication channel, and the used protocol as well to get reliable results.

In our simulation we modeled an **attack set** of simple routing attacks such as replay, drop, and insertion attacks, as well as a number of security attacks that can be used for Man-in-the-Middle attacks. The latter are performed by combining specific routing attacks with security attacks. Our set of security attacks includes bit flips, byte exchanges, and modifications of entire data fields since each of these attacks can indicate a yet unknown attack that can be related to a Man-in-the-Middle attack. This set of attacker models is not conclusive and will be extended over time.

IV. IDS Implementation

In this section we discuss the CoAP integration of three well-known general IDS approaches, namely rule based IDS, anomaly IDS and machine learning approaches. In Subsection A of this section we first outline the IDSs and in Subsection B we give a short introduction on the integration of the three approaches for CoAP. A more comprehensive description of the IDS algorithms and their integration possibilities is out of the scope of this paper.

A. Common Intrusion Detection Approaches

Rule based approaches use predefined rules (or signatures) to separate wanted network behavior from unwanted traffic. Rules can be easily formulated to clearly match wanted or unwanted protocol or application behavior. Therefore, rule based IDS produce few false alarms. Rule based approaches have the disadvantage of not being able to detect zero-day attacks but they can be lightweight and very effective against known attacks. As new attacks appear, the given rule set has to be updated, which can lead to an increased maintenance effort.

We use a set of user defined simple protocol and application dependent rules for our simulation.

Anomaly based approaches use a mathematical model to distinguish wanted from unwanted traffic. The mathematical model represents a mapping of wanted or known traffic. The network traffic is applied to the model as a byte stream and the result is compared to the result that wanted traffic would produce. All deviations in the results are regarded as an anomaly. In most cases anomaly based approaches draw more computing power than rule based approaches but they can detect zero-day attacks. Anomaly based approaches do not need maintenance but an initial training phase or an initial parameter set to work. In addition, anomaly IDS are able to learn since the mathematical model can be adapted to changing traffic conditions. In this work we use parts of the cluster algorithm that has been proposed by Rajasegarar [14] as anomaly based approach.

Machine learning approaches implement some kind of artificial intelligence to decide about wanted and unwanted network traffic. The traffic passes a decision network that results in a rejection or acceptance. As the name already suggests these approaches are able to learn which reduces the maintenance effort but they need an initial training phase or a given set of parameters to work. We focus on the implementation of a hand tuned neural network. Neural networks are a subgroup of machine learning approaches. We examine a network of McCulloch-Pitts cells [15] for the public transport scenario since these cells are able to express logic functions and even represents the simplest form of a neural cell.

B. General IDS Integration in CoAP

IDSs, most generally, analyze data and communication patterns to identify potential attacks. Since an IDS shall scan the network traffic of a system it needs access to its communication data. As described in Section III.B the IDS can access this data using the hardware abstraction layer which is provided by the real nodes and the simulation environment. Depending on the kind of intrusion detection system it either works on a raw byte or bit stream as input or on the data fields of a message. The HAL in our simulation transmits the messages as a raw byte stream to the IDS. Therefore, the byte stream has to be adapted by the IDS to its needs.

Anomaly based IDSs and most **machine learning approaches** work on a raw byte stream and just check the different bytes to fit in a known mathematical model of the network traffic. Neuronal networks, for example, represent a machine learning approach that works on raw bit streams. In this case the IDS has to generate a bit stream out of the byte stream first.

Rule based IDSs work on processed messages since they need to access the data fields inside the messages to check them against predefined rules. In this case the byte stream first has to be processed to fit in a message structure so that the data fields inside the message can be accessed. CoAP consists of a 4 byte header, followed by a variable-length token (0 to 8 bytes), any arising CoAP options and the actual payload:

IP || UDP || CoAP (header || token || options || payload)

The CoAP header consists of a 2 bit version field, a 2 bit message type field, a 4 bit token length field, a 1 byte message code field, and a 2 byte message ID field.

V. Evaluation

In this section we demonstrate how to apply the proposed framework for the example scenario introduced in Section II. Therefore, we outline the required setup of the simulation and the hardware platform in Subsection A and present results of the security analysis of the three IDSs in Subsection B.

A. Integration of the Public Transport Example

We integrated the introduced example following the flow introduced in Figure 2. Therefore, we applied the existing CoAP code and the IDSs from the IDS library.

The application code is implemented in the central control station and the node modules. The control station sends CoAP requests to the nodes. The nodes process the requests messages and answer with CoAP responses. The code generation step assembled the C/C++ code for the system. For practical evaluation, we compiled this code for our test environment. The current in-lab test bed contains 6 real nodes (Raspberry Pi B) and a border router / control station (Linux Notebook). The nodes use either Bluetooth or WLAN modules for their communication. This small practical setup could validate the implementation but does not allow us to test the IDSs sufficiently. For extended evaluation we applied the system code to the OMNeT++ simulation. The simulation contains the application scenario, i.e. the topology of the example scenario, devices and connections between the devices, parts of the application logic, and attacker models to simulate attacks.

The application scenario and topology is implemented in OMNeT++ using modules that represent different devices or messages. For the public transport example scenario we modelled a central control station (CCS), a border router (BR), nodes (N), an attacker (A), as well as CoAP request and response messages.

The attacker models are implemented in the attacker that acts similar to the nodes, i.e. it sends and receives messages but is malicious and disturbs the network operation by applying the attacker models to replay, insert, and modify messages in the network. The executed attacks were introduced in Section III.C.

B. Results

The result of the integration is a hardware testbed and a simulation setup, both implementing the smart transport scenario. While the testbed provides general functional validation of the application, protocols and IDSs, we applied the simulation to analyze the quality of different IDS approaches. Table I shows first results of our simulation with different intrusion detection systems. We applied simple routing attacks such as message insertion, drop, and replay as well as attacks

that combine a routing attack (RA) and security attacks such as bit flip, byte exchange, or data field change.

As we can derive from the first simulation results, the rule based approach is limited by the possibility to express complex behavior in rules. On the other hand it is working well with exactly defined behavior and detected all of the routing related attacks by not being able to detect most of the security related attacks. Both the anomaly algorithm and the neural network approach were able to detect most of the attacks that were applied by us.

All IDSs can be further tuned to fulfil their task better but in general it is not possible to detect most attacks with the rule based approach due to the mentioned limitations. Therefore, we suggest to use a combination of a rule based IDS and an anomaly based or NN based approach. That way the detection efficiency for known attacks will be high but new attacks can also be detected.

TABLE I. APPLIED ATTACKS THAT WERE DETECTED

Attack	Rule IDS	Anomaly IDS	NN IDS
Insertion attack	x	x	x
Replay attack	x	x	x
Drop attack	x	x	x
Bit flip + RA	partly	x	x
Byte change + RA	partly	x	x
Field change + RA	partly	partly	partly

VI. CONCLUSIONS AND FUTURE WORK

The results presented in the previous section show two important conclusions: first, intrusion and anomaly detection is feasible for smart city applications that use the Constrained Application Protocol (CoAP). We have demonstrated three of such IDS implementations in a simulation and in practice. The second conclusion is that a hybrid IDS approach is a favorable candidate for the discussed application scenarios, since it shows the best detection rate while still having a reasonable footprint.

The results in this paper were facilitated by the novel software synthesis and simulation framework. The framework allowed us to integrate the same application, network and IDS code for both, a real hardware platform, and a complex simulation to systematically test the quality of attack and anomaly detection.

Beside an improved quantification of the simulated results, for future work it is intended to apply the presented framework to tune IDS approaches for a given scenario to identify preferable cost-efficient IDSs with the best detection quality.

ACKNOWLEDGMENT

This work has been partially funded by the European Commission under the 7th Framework Programme (FP7-SMARTCITIES-2013) in the project SMARTIE and was supported in part by the National Science Foundation under NSF grant number 1136146.

REFERENCES

[1] http://tools.ietf.org/html/draft-ietf-core-coap-18/. "Constrained Application Protocol (CoAP) draft-ietf-core-coap-18." 2013.

[2] L. Wallgren, S. Raza, and T. Voigt. "Routing Attacks and Countermeasures in the RPL-based Internet of Things." International Journal of Distributed Sensor Networks, 2013.

[3] http://inet.omnetpp.org/. "OMNeT++ simulation framework for building network simulators." 2014.

[4] N. Ferguson and B. Schneier. "A cryptographic evaluation of IPsec." Counterpane Internet Security, Inc 3031. 2000.

[5] N. J. Al Fardan and K. G. Paterson. "Lucky thirteen: Breaking the TLS and DTLS record protocols." Security and Privacy (SP), 2013 IEEE Symposium on. IEEE, 2013.

[6] A. Abduvaliyev, A. S. K. Pathan, J. Zhou, R. Roman, W. C. Wong. "On the Vital Areas of Intrusion Detection Systems in Wireless Sensor Networks." IEEE Communications Surveys and Tutorials 15.3, p. 1223-1237, 2013.

[7] I. Butun, S. Morgera, and R. Sankar. "A Survey of Intrusion Detection Systems in Wireless Sensor Networks." p. 1-17, 2013.

[8] N. A. Alrajeh, S. Khan, and B. Shams. "Intrusion detection systems in wireless sensor networks: a review." *International Journal of Distributed Sensor Networks*, 2013.

[9] A.P. Castellani, M. Rossi, and M. Zorzi. "Back pressure congestion control for CoAP/6LoWPAN networks." Ad Hoc Networks 18, p. 71-84, 2014.

[10] P. Kasinathan, C. Pastrone, M. A. Spirito, M. Vinkovits. "Denial-of-Service detection in 6LoWPAN based internet of things." Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on. IEEE, 2013.

[11] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, M. A. Spirito. "DEMO: An IDS framework for internet of things empowered by 6LoWPAN." Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013.

[12] S. Raza, L. Wallgren, and T. Voigt. "SVELTE: Real-time intrusion detection in the Internet of Things." *Ad hoc networks* 11.8, p. 2661-2674, 2013.

[13] http://inet.omnetpp.org/. "INET Framework." 2014.

[14] S. Rajasegarar, C. Leckie, and M. Palaniswami. "Hyperspherical cluster based distributed anomaly detection in wireless sensor networks." Journal of Parallel and Distributed Computing 74.1, p. 1833-1847, 2014.

[15] W. S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." The bulletin of mathematical biophysics 5.4, p. 115-133, 1943.