Applying Intrusion Detection Systems to Wireless Sensor Networks

Rodrigo Roman
E.T.S. Ing. Informatica
University of Malaga
29071, Malaga, Spain
roman@lcc.uma.es

Jianying Zhou
Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
jyzhou@i2r.a-star.edu.sg

Javier Lopez
E.T.S. Ing. Informatica
University of Malaga
29071, Malaga, Spain
jlm@lcc.uma.es

Abstract—**The research of Intrusion Detection Systems (IDS) is a mature area in wired networks, and has also attracted many attentions in wireless ad hoc networks recently. Nevertheless, there is no previous work reported in the literature about IDS architectures in wireless sensor networks. In this paper, we discuss the general guidelines for applying IDS to static sensor networks, and introduce a novel technique to optimally watch over the communications of the sensors' neighborhood on certain scenarios.**

Keywords: Internet security, intrusion detection systems, wireless sensor networks.

I. Introduction

An intrusion can be defined as a set of actions that can lead to an unauthorized access or alteration of a certain system. The task of Intrusion Detection Systems (IDS) is to monitor computer networks and systems, detecting possible intrusions in the network, and alerting users after intrusions had been detected, reconfiguring the network if this is possible [1].

Although there have been some recent developments in the area of IDS systems for wireless ad hoc networks, there is no previous work reported in the literature about IDS architectures for wireless sensor networks. The purpose of this paper is to show why IDS solutions created for ad hoc wireless networks cannot be applied directly to sensor networks, and introduce the general guidelines for applying IDS architectures in static sensor networks (with no mobile nodes). Also, a novel technique for optimally monitoring neighbors, that we have called *spontaneous watchdogs*, is introduced.

The rest of the paper is organized as follows. Section II shows and compares the state-of-the-art of Intrusion Detection Systems for ad hoc networks and sensor networks. Section III presents the general guidelines of an IDS architecture for sensor networks. Section IV concludes the paper, discussing the future work and open research areas.

II. IDS and Wireless Networks

A wireless network consists of a collection of nodes that are able to maintain a wireless communication channel between each other without relying on any fixed infrastructure. This and other factors make wireless networks substantially different from wired networks: there is no single point of access to the wireless network, all nodes must cooperate to create the network and set up its infrastructure (e.g. routing subsystem) and services (e.g. data collection), and finally, most operations are carried out in a distributed manner.

Both ad hoc networks and sensor networks can be included inside the wireless networks category. However, there are some important differences between them:

- In ad hoc networks, every node is usually held and managed by a human user. However, in a sensor network, every node is totally independent, sending data and receiving control packets from a central system called *base station*, usually managed by a human user.
- Computing resources and batteries are more constrained in sensor nodes than in ad hoc nodes. A typical sensor node such as MICA2 [6] has a 8Mhz microprocessor with 128Kb of program flash memory and 512Kb of serial flash memory.
- The purpose of sensor networks is very specific: measure the physical information (such as temperature, sound, ...) of its surroundings. As a result, both hardware modules and communication/configuration protocols are highly specialized.
- Node density in sensor networks is higher than in ad hoc networks. However, sensor nodes have more chances to fail and disappear from the network, due to the battery constraints and the low physical security.

As a result of these differences, IDS solutions developed for ad hoc networks [7], [8], [9], [10] cannot be applied directly to sensor networks. First, it is not possible to have an active full-powered agent inside every node. Also, an IDS for sensor networks must send the alerts to the base station in order to warn the human user. Finally, the IDS must be simple and highly specialized for reacting against specific sensor network threats and to the specific protocols used over the network.

Partial solutions exist that allows a node to check the security of the sensor network, by testing whether a group of nodes are alive or dead [3], analyzing fluctuations in sensor readings [4], attesting the integrity of the code inside a node [5], or watching over the information interchange [11]. Nevertheless, no solution is designed specifically to interact with other schemes, although they can be used as a part of an Intrusion Detection System.

III. A General IDS Architecture for Sensor Networks

A. Detection Entities

The constraints inherent to sensor networks, such as sparse resources and limited battery life, impose a cautious planning on how the detection tasks are performed. As in ad hoc networks, IDS agents must be located in every node. However, for the sake of performance, the architecture of these agents must be divided into two parts: local agents and global agents.

- *Local agents* should monitor the local activities and the information sent and received by the sensor. This is only carried out when the sensor is active, and the sensor only manages its own communications. Thus, the overheads imposed on the sensor node are low.
- *Global agents* should watch over the communications of their neighbors, and can also behave as watchdogs [11]. However, not all nodes can perform this operation at the same time, because this operation would require sensors to analyze the contents of all packets in their radio range. Therefore, only a certain subset of the nodes must watch over the network communications at a time.

Once any agent, global or local, discovers a possible breach of security in the network, it must create and send an alert to the user. The only way the user can be reachable is through the base station. Hence, all alerts must be sent to the base station. The mechanism for sending the alerts to the base station depends on the underlying architecture of the sensor network, but it must assure that all alerts reach its destination safely (using mechanisms such as μTesla [12]).

B. Data Structures

Every agent, and hence every node, must store information about its surroundings in order to work properly. This information can be divided into two categories: knowledge about the security (an alert database that contains information about alerts and suspicious nodes), and knowledge about the environment (a list of the neighbors of the immediate neighbors of the node, which can be updated over the lifetime of the node using the received messages).

Every node has an internal alert database, which is used for storing the security information generated by the node agents. The format and size of that database is implementation-dependant (i.e., depends on the protocols used in the sensor network). Nevertheless, it must contain the following fields: time of creation [1], classification, and source of the alert (cf. [14]).

This neighbor list can be obtained "a priori", if any deployment information is available, or after the deployment, using the same assumption of the LEAP [15] protocol (the network will be secure in the first t seconds after the deployment). One problem of this list is its memory footprint: The list grows as a quadratic function $((n^2) + n)$ of the number of one-hop neighbors, hence it is not scalable for high density networks. However, the size of the list can be reduced using Bloom filters [16], storing the neighbors' list of every neighbor as a bit array. For a configuration of $k = 1$ (hash functions) and $m/n = 2$ (number of bits doubles the number of neighbors), the size of the list is reduced 75%, at the cost of introducing a probability between 16% and 40% of false positives.

[1]Time synchronization between nodes, albeit an important issue on alert management, is out of scope of this paper. (It has been discussed in [13].)

C. Local Agents

The task of local agents is to discover any attack or threat that can affect the normal behavior of the sensor nodes by analyzing only the local sources of information. Those sources are the actual status of the node, packets received and sent by the node, the measurements made to the environment, and all the available information about its neighbors.

What kind of attacks should be detected by the local agent? First, attacks against the physical or logical safety of sensor nodes can be discovered if the nodes are able to know whether they are being manipulated or not. Sensor nodes are able, for example, to detect whether they are being reprogrammed [2], so they can raise an alarm before allowing the execution of any new code.

Node measurements are also vulnerable. Since the primary task of the sensor nodes is to analyze environmental data, any adversary can try to influence this process for his own benefit. Nevertheless, all data being read by the nodes come from the real world, and follow certain patterns and limits. Therefore, anomaly detection techniques can be used for monitoring these measurements. For example, if the node is going to be deployed in a static place, any variation in the accelerometer means that the node is being taken by an unknown source, so it will raise an alarm.

Finally, the local agent also monitors packets which are addressed directly to the node, although developing a lightweight detection technique for every existent protocol is out of scope of this paper because of large number of protocols and packet formats present in the literature (e.g., routing algorithms [17]). However, there are some issues related to local-processed packets that are protocol-independent and can be discussed here: the incorporation of a new node into the network and jamming of the signal.

In static scenarios, where few nodes are added after the initial deployment, every node can take advantage of the list of known neighbors. Every time a node receives a packet from a new neighbor, it will add it to the list, and raise an alarm. If human users of the network know that they didn't include any node into the network, they will be aware that the new node belongs to an adversary. Also, if a node tries many times to send a packet but the channel is not available, misuse techniques can be employed to detect if this is an abnormal situation that must raise an alarm.

[2]Using, for example, the Xnp component of the TinyOS operative system.

D. Global Agents

As stated before, global agents must be in charge of analyzing packets that their immediate neighbors send and receive. They can also behave as watchdogs [11] receiving and processing the packets relayed by next-hop nodes using protocol-dependant techniques. Since global agents are able to receive packets from both the neighbors and relayed by the next-hop (due to the broadcast nature of the communications), they can be prepared to detect whether a certain node is dropping or modifying packets by analyzing those packets.

However, if all global agents are activated and listening to their neighborhoods at the same time, analyzing the network would be a costly operation in terms of energy. As a result, only a certain subset of nodes that cover all communications in the sensor network should activate their global agents. How this task is done depends on the underlying architecture of the sensor network. There are two basic architectures that specify how the sensors route the information over the network and how sensors group themselves. These two architectures are called hierarchical and flat.

In *hierarchical* configurations, sensors are grouped into clusters. One of the members of the cluster behaves as server, or "cluster head" (which can be more powerful than the other nodes [18] or not [19]), with management and routing tasks. On the other side, in *flat* configurations, information is routed sensor by sensor (every sensor of the network participates in the routing protocol), and almost all sensors have the same computational capabilities and constraints.

In hierarchical architectures, global agents are activated in every cluster head, because the combination of all cluster heads covers (in most cases) the entire sensor network. Consequently, total network coverage is assured. This approach helps to preserve the overall energy of the system because cluster heads are either more powerful than other nodes or are rotated periodically.

Activation of global agents is more difficult to manage in flat architectures because it is not possible, a priori, to know what nodes can cover the entire network. One possible solution is to use clustering techniques only for the Intrusion Detection System [20], where more powerful or tamper-protected cluster heads will be periodically elected with the only purpose of activating their global agents in order to cover the interchange of messages over the network.

This clustering solution adds some complexity to the network in the creation and maintenance of clusters with maximum global coverage, adding a possible point of attack and the overhead of periodical control messages. However, there is an alternative distributed solution that can be applied to flat architectures without organizing them into clusters or adding more powerful nodes, which is called *spontaneous watchdogs*.

E. Spontaneous Watchdogs

The spontaneous watchdog technique relies on the broadcast nature of sensor communications, and takes advantage of the high density of sensors being deployed in the field. For every packet circulating in the network, there are a set of nodes that are able to receive both that packet and the relayed packet by the next-hop, as shown in Fig. 1. Hence, all these nodes have a chance to activate their global agents in order to monitor those packets. The main goal of our solution is to activate only one global agent per packet circulating in the network. The process is as follows:

- Every active node will receive all packets sent inside its neighborhood, due to the broadcast nature of communications.
- The node will check if it is the destination of the packet. If not, it will not drop the packet instantly. Instead, it will check if the destination of the packet is in its neighborhood (thus it could receive any packet forwarded by the destination). This check can be done because all nodes store a list of neighbors for every node in its neighborhood.
- If true, the node can be a spontaneous watchdog. Consequently, it will calculate how many nodes in the network are in its same situation.
- If the number of nodes that fulfill the requirements are n, a single node will select itself as a global agent for this packet with a probability of $\frac{1}{n}$. This process can resemble as n people with 1 dice of m sides each, where $n = m$, trying to obtain a 1 in the dice to activate the global agent.

Fig. 1. Possible Spontaneous Watchdogs

It could seem that this technique increases the energy consumption of the nodes to prohibitive levels by making them receive and analyze every packet sent inside their neighborhood. However, in the MICA2 radio stack, every node must receive and process the packets sent by their neighbors - they cannot know if the packet is addressed or not to them "a priori" [21]. Therefore, the only overhead imposed to the nodes will be the decision of being an spontaneous watchdog, and the energy consumption of activating the global agent will be distributed over all of them.

The spontaneous watchdog technique, however, does not assure that one and only one node will activate its global agent for every packet in the network, due to the independence of the nodes' behavior. The probability that α nodes have to activate

Fig. 2. Number of Spontaneous Watchdogs, with (a) Normal Probability, (b) Double Probability

their global agents at the same time is given by the following equation:

$$f(\alpha, n, m) = \frac{PR_n^{n-\alpha,\alpha} \cdot (m-1)^{n-\alpha}}{VR_{m,n}} \quad (1)$$

where $PR_n^{n-\alpha,\alpha}$ is the formula of permutation with repeated elements, $VR_{m,n}$ is the formula of r-permutations with repetition, *n* is the number of nodes that could activate their global agents (i.e., the number of nodes that are going to throw a dice), and *m* is the number of nodes that are going to influence on the probability of activating the global agents, normally equal to n (i.e., the number of sides of every dice).

In the assumption that every node has the same probability of activating its global agent (as shown in Fig. 2a, where $n = m = \{3, 5, 10, 25\}$ and $\alpha = [0..10]$), one of three packets in the network will go unsupervised, and most packets will be analyzed by one or two nodes of their neighborhood independently of the density of the network.

However, it is possible to tweak the behavior of the nodes by increasing that probability (decreasing *m* in Eq. (1)), as shown in Fig. 2b. Therefore, it is possible to decrease the number of packets left unsupervised by increasing the probability of a single node being a watchdog, but at the same time the number of nodes activating their global agents will increase.

F. Agent Cooperation

As stated before, both local and global agents reside in the same node, thus the results of their observations are stored in a single alert database. When an agent (local or global) wants to access information about previous actions of a suspicious node, it can take advantage of the results obtained by itself and the other agent. As a result, a collaboration between global and local agents in the same node is achieved.

However, in most situations (e.g. a global agent in a spontaneous watchdog scenario) an agent will have to collaborate with agents located in other nodes in order to obtain a more accurate description of a threat, obtain more information about a certain node, or aggregate their alert databases. Therefore, an agent must be able to securely communicate and interchange information with its neighbors using schemes such as majority voting schemes [7], although this exchange should be reduced to simple reports (e.g. about the credibility of a node) due to bandwidth and energy constraints. Secure and authenticated communications between neighbors are easily achieved in a sensor network environment using existent key management schemes [22].

Since any sensor node is prone to be tampered (or its identification falsified), any report coming from a single node must be weighted with the information provided by the other nodes. Local agents must be also aware of nodes that repeatedly ask for information about other nodes, since they can be rogue nodes trying to make a sleep deprivation attack.

IV. Conclusion

In this paper, we studied and discussed why IDS architectures for ad hoc networks cannot be applied into a sensor network scenario. We also proposed a general IDS architecture for static sensor networks, and introduced a new technique, the spontaneous watchdogs, where some nodes are able to choose independently to monitor the communications in their neighborhood.

Future work will involve the implementation and simulation of the architecture over a particular group of protocols in order to study the energy consumption and IDS performance of this model. There are other factors that must be thoroughly investigated, such as how a node can deduce the number of neighbors that can activate their global agents if no additional information is available (e.g. when nodes cannot store the complete neighbors list), how real-life radio models can affect the spontaneous watchdog technique, how to successfully aggregate alert data in a flat network, and other issues.

References

[1] R. Bace. *Intrusion Detection*. MacMillan Technical Publishing, 2000.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. *Wireless Sensor Networks: A Survey*. Computer Networks, 38(4), March 2002.

[3] C. Hsin and M. Liu. *A Distributed Monitoring Mechanism for Wireless Sensor Networks*. 1st ACM Workshop on Wireless Security (WiSe'02), September 2002.

[4] S. S. Doumit and D. P. Agrawal. *Self-Organized Criticaly & Stochastic Learning Based Intrusion Detection System for Wireless Sensor Networks*. 2003 Military Communications Conference (MILCOM'03), October 2003.

[5] A. Seshandri, A. Perrig, L. Van Doorn, and P. Khosla. *SWATT: SoftWare-based ATTestation for Embedded Devices*. 2004 IEEE Symposium on Security and Privacy, May 2004.

[6] Crossbow Technology, Inc. *MICA2, Wireless Measurement System*. http://www.xbow.com.

[7] Y. Zhang and W. Lee. *Intrusion Detection Techniques for Mobile Wireless Networks*. ACM/Kluwer Wireless Networks Journal, 9(5):545-556, September 2003.

[8] J. Kong, H. Luo, K. Xu, D. L. Gu, M. Gerla, and S. Lu. *Adaptive Security for Multi-Layer Ad-Hoc Networks*. Special Issue of Wireless Communications and Mobile Computing, 2002.

[9] P. Albers, O. Camp, J. Percher, B. Jouga, L. Me, and R. Puttini. *Security in ad hoc Networks: A General Intrusion Detection Architecture Enhancing Trust Based Approaches*. 1st International Workshop on Wireless Information Systems (WIS'02), April 2002.

[10] C. Karlof and D. Wagner. *Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures*. 1st IEEE International Workshop on Sensor Network Protocols and Applications, May 2003.

[11] S. Marti, T. Giuli, K. Lai, and M. Baker. *Mitigating Routing Misbehavior in Mobile Ad Hoc Networks*. 6th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), August 2000.

[12] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. *SPINS: Security Protocols for Sensor Networks*. 7th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01), July 2001.

[13] S. Ganeriwal, S. Capkun, C. Han, and M. Srivastava. *Secure Time Synchronization Service for Sensor Networks*. 4th ACM Workshop on Wireless Security (WiSe'05), September 2005.

[14] H. Debar, D. Curry, and B. Feinstein. *The Intrusion Detection Message Exchange Format*. draft-ietf-idwg-idmef-xml-14 (Work in Progress), January 2005.

[15] S. Zhu, S. Setia, and S. Jajodia. *LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks*. 10th ACM Conference on Computer and Communications Security (CCS '03), Washington D.C., October 2003.

[16] B. Bloom. *Space/Time Trade-offs in Hash Coding with Allowable Errors*. Communications of the ACM, 13(7):422-426. July 1970.

[17] J. N. Al-Karaki and A. E. Kamal. *Routing Techniques in Wireless Sensor Networks: A Survey*. IEEE Wireless Communications, 11(6):6-28. December 2004.

[18] EYES European Research Project, IST-2001-34734. http://eyes.eu.org.

[19] O. Younis and S. Fahmy. *Distributed Clustering in Ad-Hoc Sensor Networks: A Hybrid, Energy-Efficient Approach*. IEEE INFOCOM'04, March 2004.

[20] F. Anjum, D. Subhadrabandhu, S. Sarkar, and R. Shetty. *On Optimal Placement of Intrusion Detection Modules in Sensor Networks*. 1st International Conference on Broadband Networks (BROADNETS'04), October 2004.

[21] MICA2 Radio Stack for TinyOS. http://www.tinyos.net/tinyos-1.x/doc/mica2radio/CC1000.html

[22] S. A. Camtepe and B. Yener. *Key Distribution Mechanisms for Wireless Sensor Networks: A Survey*. TR-05-07 Rensselaer Polytechnic Institute, Computer Science Department, March 2005.

APPENDIX
PROOF OF EQUATION (1)

The main purpose of (1) is to know the probability that α nodes have to activate their global agents at the same time in a neighborhood of *n* possible global agents, when the decision of a node cannot influence others. This problem can be abstracted into another one:

We have n players (nodes) in a table, and every player has a dice with m sides (where m is usually equal to n). We want to know the probability of α players obtaining a "1" (i.e., activating the global agents) when all players have thrown their dices once.

In this new problem, we want to solve:

$$f(\alpha, n, m) = \frac{f'(\alpha, n, m)}{f''(\alpha, n, m)} \tag{2}$$

where $f'(\alpha, n, m)$ are all cases where *n* dices of *m* sides are thrown and α and only α dices have a result of 1; $f''(\alpha, n, m)$ are all cases where *n* dices of *m* sides are thrown.

If a player throws a dice, the result can be within one of these two sets: $\{1\}$ or $\{2..\text{m}\}$. The number of possible cases where α and only α dices fall in the $\{1\}$ set is the number of permutations with repetition [3] of *n* elements (dices) where the value that α elements fall into the $\{1\}$ set and $n-\alpha$ elements fall into the $\{2..\text{m}\}$ set is

$$PR_n^{n-\alpha,\alpha} = \frac{n!}{n-\alpha! \cdot \alpha!} \tag{3}$$

With (3), we have obtained the number of cases where α and only α dices fall in the $\{1\}$ set. For example, if we have $n = 2$ dices with $m = 3$ sides each, the number of cases where one and only one dice ($\alpha = 1$) falls in the $\{1\}$ set are equal to 2, i.e., $\{$(1,[2..3]), ([2..3],1)$\}$.

For every case in the previous example, we have $n - \alpha$ dices that fall in the $\{2..\text{m}\}$ set, with $m-1$ elements inside. Therefore, it is possible to know the number of cases where α and only α dices have a result of 1, or $f'(\alpha, n, m)$, if we multiply (3) with $(m-1)^{n-\alpha}$:

$$f'(\alpha, n, m) = PR_n^{n-\alpha,\alpha} \cdot (m-1)^{n-\alpha} \tag{4}$$

because $(m-1)^{n-\alpha}$ are the ordered variations of $m-1$ elements taken $n-\alpha$ times with repetition. In the previous example (n = 2, m = 3, α = 1), the number of cases where one and only one dice has a result of 1 are $2 \cdot 2 = 4$, i.e., $\{$(1,2), (1,3), (2,1), (3,1)$\}$.

Finally, $f''(\alpha, n, m)$, or all cases where *n* dices of *m* sides are thrown, is the r-permutations with repetition of m elements taken n times:

$$f''(\alpha, n, m) = VR_{m,n} = m^n \tag{5}$$

We can conclude that both (1) and (2) are equal, and both solve the same problem.

$\square$

[3] We use permutations here because the order of the dices is important, since every dice (node probability) is different and independent from the others.