On Simulative Analysis of Attack Impact in Wireless Sensor Networks

Gianluca Dini and Marco Tiloca
Dipartimento di Ingegneria dell'Informazione
University of Pisa, Pisa, Italy
{gianluca.dini, marco.tiloca}@iet.unipi.it

Abstract

Wireless Sensor Networks (WSNs) are frequently adopted in industrial applications. However, they are particularly prone to cyber-physical attacks. Since addressing all possible attacks is not viable, due to performance and economic reasons, it is vital to choose which attacks to address and which countermeasures to adopt. Hence, a quantitative analysis of attack impact is crucial to make an effective choice. In this paper, we present a simulative approach to attack impact analysis, and show that simulation results provide valuable insights on the attack severity. To fix ideas, we refer to a WSN monitoring pollutant emissions of a critical infrastructure. We analyze effects of cyber-physical attacks against the network, and rank them according to their impact severity. This supports designers in deciding which attacks to address and which countermeasures to select.

1. Introduction

In the recent years, Wireless Sensor Networks (WSNs) have been widely employed as part of many industrial communication systems [15]. However, WSNs are also particularly prone to a great number of security attacks [6], and security infringements in industrial systems may cause losses, damages and even injuries (e.g. environmental pollution). It follows that it is vital to properly protect WSNs by adopting appropriate security countermeasures.

However, it is well known that achieving perfect security is not possible, for both performance and economic reasons [2]. In fact, this might result in excessive costs and performance penalties, which could be unaffordable especially in resource-scarce devices as sensor nodes [2][9]. Thus, it is vital to clearly define a *threat model*, and then perform *risk assessment*, in order to determine the extent of potential threats and identify appropriate solutions [7]. A *risk* is defined as a function of the likelihood of a given threat to occur (*feasibility*), and its resulting consequences (*impact*). Hereafter, we focus on the attack impact.

In this paper, we discuss a simulative approach to attack impact analysis, based on the following three steps. First, we evaluate the effects of attacks on the network and the application through simulation. This allows for understanding if a given attack is effective, and adopting solutions against it is actually required. Second, we rely on simulation results and a set of *security metrics* in order to quantitatively evaluate the impact of attacks, and rank them according to their severity. This makes it possible to understand which security attacks deserve more attention than others, and thus establish a priority among threats and attacks. Third, and finally, we rely on simulation to evaluate different security solutions against considered attacks. This allows a designer to compare their effectiveness and efficiency, and choose the most appropriate ones.

We claim that our approach is particularly effective, due to the following reasons. It does not require a real deployed network, thus allowing for thoroughly performing the attack impact analysis during the design phase. Besides, it allows designers to evaluate attack impact in a quantitative way, thus achieving a measurement of impacts' magnitude which can be used in the cost-benefit analysis of possible solutions [7]. Finally, relying on simulation makes it possible to consider, even simultaneously, both attacks against the physical system, as well as cyber attacks performed at different network levels (e.g. application, routing protocols, medium access protocols).

In order to support our points, we refer to a pollution monitoring application and a possible threat model as a case study. A WSN deployed in the field monitors pollution levels and detects infringements of pollution limitations. We consider an adversary performing both physical and cyber attacks in order to conceil irregular emission rates, and rely on our *ASF* attack simulation framework [5] to evaluate attack effects by means of simulation. We show that our simulation results provide us with valuable insights on the attack impact, and discuss how it is possible to devise appropriate countermeasures to be adopted.

Other approaches to attack impact analysis have been presented. Analytical models aimed at detecting and contrasting attacks are discussed in [12][17][20], and simulation is used to validate their correctness and efficiency. In [19], the authors present *SenSec*, a framework that simulates occurrence of attacks by injecting events into real application simulators. Unlike *SenSec*, our framework *ASF* assumes that attacks have been successfully performed, and reproduces their effects on the network and application, rather than their actual performance. Finally, a different approach relies on system-theoretic solutions. Cárdenas *et al.* focus on process control systems, and stress the importance of incorporating the knowledge of the physical system under control [1]. Mo *et al.* consider Smart Grid infrastructures, and claim both information security and system-theory-based security are essential to secure cyber-physical systems [18].

The rest of the paper is organized as follows. In Section 2, we describe the application scenario we refer to. Section 3 presents the considered threat model, while Section 4 recalls the *ASF* framework. Section 5 discusses attack simulation results. In Section 6, we define three attack metrics, and rank considered attacks according to their severity. Finally, Section 7 evaluates different security solutions, while, in Section 8, we draw our conclusions.

2. Application scenario

In this paper, we consider an environmental monitoring WSN application, aimed at measuring pollution levels within an industrial area. In particular, we refer to the industrial field depicted in Figure 1, where three independent plants release pollutant into the air, through smokestack *S1*, *S2*, and *S3*, respectively. In addition, a WSN has been deployed in the field, in order to monitor pollution levels and detect possible misbehaviors.

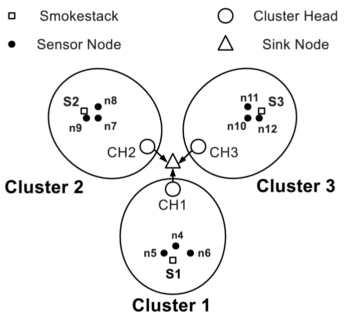

Figure 1. Application scenario.

More in detail, each smokestack is associated with one *Cluster*, i.e. *C1*, *C2*, and *C3*, respectively. Each Cluster includes one *Cluster Head* node, i.e. *CH1*, *CH2*, and *CH3*, respectively, and three sensor nodes. Finally, the network comprises also one *Sink* node. Every sensor node periodically senses pollution emission from the smokestack in the associated Cluster, and sends its report to the associated Cluster Head node. The latter periodically computes an average pollution level according to received reports, and delivers it to the Sink node. Finally, the Sink node checks whether any report exceeds a given threshold *T*.

The Sink node also aggregates average reports received by Cluster Head nodes, to detect possible infringements of pollution limits from a whole field stanpoint. So doing, possible anomalies, malfunctionings, or even conscious illegal deeds can be signaled to a centralized control system. Then, the latter would be able to restore normal operating conditions, and prevent more severe consequences, such as injury to people or environmental disasters. In the following, we focus on monitoring operations and communication among sensor nodes, while control and recovery procedures are out of the scope of this paper.

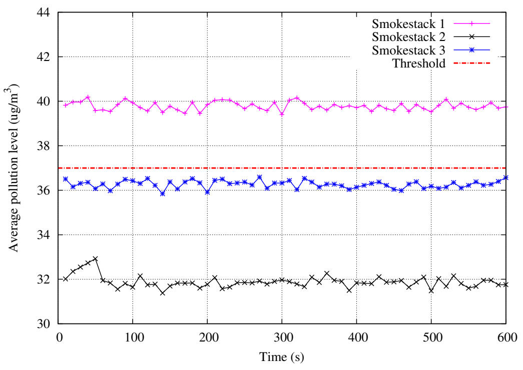

Figure 2. Pollution level.

In our case study, we assume that the plant located in Cluster *C1* has been maliciously altered, in order to conceil breaches of pollution limits. That is, as shown in Figure 2, smokestack *S1* infringes the pollution level limit, that we have supposed to be fixed at 37 $\mu g/m^3$. The dashed line depicts the pollution level that smokestacks are supposed to not exceed. The other curves represent the average pollution levels over time for smokestack *S1*, *S2*, and *S3*. The graph shows that emissions from *S1* are exceeding the defined threshold, thus an urgent intervention is required in order to restore normal operating conditions.

Thanks to the monitoring WSN, it is possible to detect anomalies in pollutant emissions and react promptly. This assumes that sensor nodes and Cluster Head nodes work correctly, i.e. collected data are genuine and report delivery occurs regularly. However, as we describe in Section 3, an adversary can perform a number of attacks against the WSN, and tamper with the data collection process, thus altering the monitoring process, and making falsely appear the behavior of smokestack *S1* as regular.

3. Threat model

With reference to the application scenario described in Section 2, an adversary may compromise service availability, by altering reports produced by sensor nodes before being collected by Cluster Head nodes. We consider an adversary interested in altering the computation of average pollution levels on Cluster Head node *CH1*. If she managed to bring average pollution levels below the fixed threshold, pollutant emissions from smokestack *S1* would appear as regular, conceiling an actual limit infringement. In the following, we consider three possible attacks against the WSN, namely injection attack, misplace attack, and wormhole attack. The first attack is purely cyber, the second one is purely physical, whereas the third one is a cyber-physical one. Therefore, this attack selection provides the full range of attack types that can be launched against the WSN.

Injection attack. The adversary creates new fake report packets, and inject them into Cluster *C1*, pretending they have been sent by a legitimate sensor node belonging to *C1*. Of course, fake values carried by such reports alter the computation of average pollution levels on Cluster Head *CH1*. This attack is quite hard to be detected. However, comparisons with other nodes' reports may help to contrast its effectiveness.

Misplace attack. The adversary captures one sensor node from Cluster *C1*, and moves it from its original position to a new one. By properly choosing the new position, e.g. far from smokestack *S1*, it is possible to alter the computation of average pollution levels on the Cluster Head node *CH1*. This attack is far more difficult to detect, since Cluster Head nodes assume that all sensor nodes' original positions remain unchanged over time.

Wormhole attack. This attack actually consists in two steps. First, the adversary captures one sensor node n from Cluster *C1*, and places it in a different Cluster, in order to make it monitor pollutant emissions from a different (regular) smokestack. Secondly, she tampers the misplaced node n [3], in order to make it perform a wormhole attack [16]. That is, node n does not send its report to the Cluster Head node in the Cluster it has been moved to. Instead, node n forwards its report to Cluster Head *CH1* through a dedicated low-latency channel. Thus, values reported by node n refer to a regular smokestack, rather then *S1*, and the computation of average pollution levels by *CH1* is obviously altered. This attack is particularly difficult to be contrasted, although some countermeasures have been proposed [4][16].

4. Attack Simulation Framework

In order to evaluate the effects of attacks described in Section 3, we relied on *ASF*, our attack simulation framework for WSNs we described in [5]. Thanks to *ASF*, the user can describe attacks, and evaluate their impact, on the network and the application. This section recalls *ASF* architecture and its main features. More details about the attack evaluation framework can be found in [5].

The *ASF* framework conceives an attack as a sequence of *events*. Two types of attacks are admitted, namely *physical attacks* and *cyber attacks*. Physical attacks consist in physical actions on sensor nodes, such as their destruction or removal. On the other hand, cyber attacks focus on the actual network communication, aiming at thwarting it by manipulating or discarding packets. *ASF* provides an *Attack Specification Language*, which allows users to easily describe attacks as a sequential list of events.

ASF evaluates attacks by means of an *Attack Simulator*. Such a component considers attack descriptions provided by the user, and simulate attacks by injecting additional simulation events at runtime. It is worth noting that the *Attack Simulator* does not reproduce the actual performance of attacks, but only their effects on the network and the application. Also, since attack events are injected at runtime, the user does not have to modify either the original application code or the communication stack on sensor nodes. The *Attack Simulator* can be realized by enhancing a discrete event network simulator. In [5], we described our prototype implementation of *ASF* for the *Castalia* simulator, which is based on the *OMNeT++* platform.

5. Attack impact analysis

In the following, we refer to the application scenario described in Section 2. Specifically, we assume that sensor nodes collect pollution measurements in their proximity one time every 70 milliseconds, while Cluster Head nodes compute average pollution levels every 10 seconds. Report packets trasmitted by sensor nodes are 39 bytes in size, and include a payload whose size is 4 bytes. Finally, the pollution level threshold is set to 37 $\mu g/m^3$. Then, we evaluate the effects of the three attacks presented in Section 3, by means of *ASF*. In our simulations, we considered the *Multipath Rings* routing protocol, the *T-MAC* MAC protocol [13], and the *CC2420* radio chipset. Results were obtained by means of 30 simulation runs, whose length was 600 seconds each. The adopted pollutant propagation model is based on the *Customizable Physical Process* provided by the Castalia simulator. In this section, we consider the attacks described in Section 3, each one of which occurs at time $t = 200$ s. In particular, we discuss the quantitative impact of each attack on the pollution monitoring process.

5.1. Injection attack

We consider an adversary injecting fake report packets into Cluster *C1*, with the aim of altering the computation of average pollution levels, on Cluster Head *CH1*. Specifically, the adversary creates fake report packets, and fills their fields as follows. First, the value 4 is written in the source node ID field of each layer header. So doing, every forged packet will appear as it has been sent from node *n4* from Cluster *C1*. Then, the report content within the application payload is set to 33 $\mu g/m^3$. Such a value is quite close to the average pollution level detected in Cluster *C2*, as shown in Figure 2. Then, fake pollution levels would result to be plausible from a Cluster Head *CH1* standpoint, so that the attack is not easy to be detected.

With reference to Cluster *C1*, we discuss the impact of the injection attack on the average pollution level computation. We consider different *injection interval* I, where $I = 70$ ms means that the adversary injects one forged report packet into Cluster *C1* every 70 ms. Figure 3 shows the effects of the injection attack for different values of I. As we can see, the larger the injection interval, the less effective the attack is. However, if I displays values smaller than 50 ms, the attack is successfully performed, and the average pollution level goes beyond the threshold. It is evident that the adversary has no reason to perform the at-

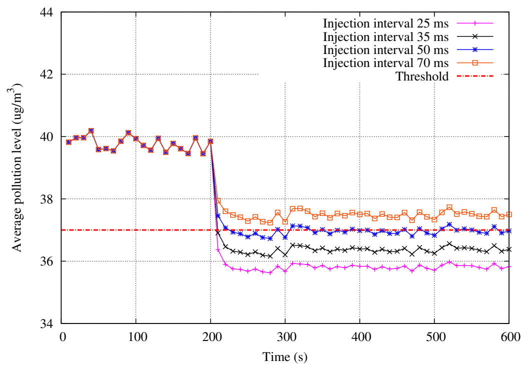

Figure 3. Pollution level (injection).

tack with an injection interval smaller than 35 ms. In fact, it would require a great energy expenditure by the adversary, and could even be perceived as a Denial of Service, with increased chance of being detected.

5.2. Misplace attack

In this attack, the adversary physically captures sensor node $n5$ in Cluster *C1*, and moves it away from smokestack *S1*. For the sake of simplicity, we assume that the misplaced node is moved along the Y-dimension only, towards Cluster Head *CH1*.

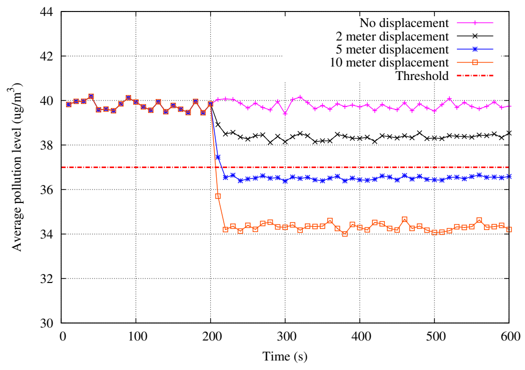

Figure 4. Pollution level (misplace).

Figure 4 shows how misplacing sensor node $n5$ affects the perceived average pollution level, considering a displacement distance *d* of either 2, 5, or 10 meters. As we can see, moving node $n5$ 2 meters away from its original position is insufficient to make average pollution level appear as regular. Instead, if $d \geq 5$ meters, the adversary manages to achieve her objective. Of course, the more far sensor nodes are misplaced, the more effective the attack is. Although we omit them for the sake of brevity, further simulative results showed us that the attack is slightly less effective if sensor node $n4$ is misplaced. Note that it might be difficult to estimate the attack impact by simply observing how sensor nodes are positioned in the field.

5.3 Wormhole attack

We consider an adversary who actually performs a combination of misplacement attack and wormhole attack. First, node *n5* from Cluster *C1* is removed from its original position, and placed near sensor node *n7*, in Cluster *C2*. Once node *n5* has been misplaced, it starts sensing pollutant emissions from smokestack *S2*, which, unlike smokestack *S1*, displays an acceptable pollution level (see Figure 2). Then, sensor node *n5* is reprogrammed, so that, for each collected sample s, it creates a perfect copy s', and sends both s and s' to Cluster Head *CH1*, through a dedicated communication channel. Since the sample interval of *n5* is 70 ms, and each sample is transmitted twice, we have a *wormhole interval* equal to 35 ms.

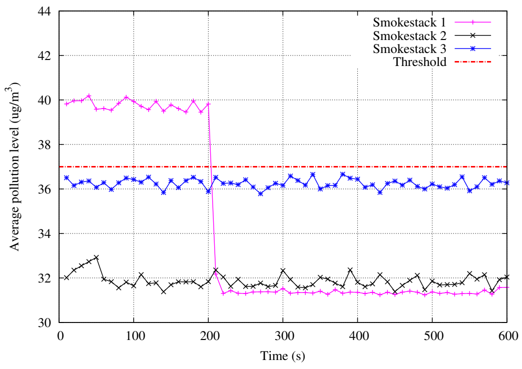

Figure 5. Pollution level (wormhole).

Figure 5 shows the effects of the wormhole attack on pollution monitoring. As we can see, the average pollution level in Cluster *C1* appears equal to about 31 $\mu g/m^3$, i.e. far below the threshold. This means the wormhole attack results to be even more effective than the injection attack discussed in Section 5.1. In fact, in case of injection attack with injection interval equal to 35 ms, Cluster *C1* displays an average pollution level comprised between 36 and 37 $\mu g/m^3$, that is closer to the threshold (see Figure 3).

6. Attack ranking

In this section, we describe a possible way to rank security attacks according to their severity. Practically, it is necessary to capture the impact of each attack, and provide a quantitative indication of its severity, with reference to *service integrity*, *network availability*, and *information confidentiality* security requirements [2]. We believe this is the first step to devise the actual security priorities and properly select the most appropriate countermeasures.

6.1. Security metrics

In the following, we define three different functions, namely *security metrics*, aimed at separately measuring how an attack A impacts against service integrity, network availability, and information confidentiality. We define such metrics as an extension of similar functions described by Cardenas *et al.* in [2].

$$x_A = \frac{\sum_{i=1}^{M} r_i \cdot ((\sum_{j=1}^{S_i} \parallel v_j - v_j' \parallel^2)/S_i)}{M} \qquad (1)$$

Equation 1 measures x_A, i.e. the level of compromise in service integrity due to attack A. That is, x_A indicates how information $v^{'}$ in the presence of attack A differs from information v when the system is attack free. More in detail, M is the amount of nodes handling forged information, whereas r_i is the weight associated to the i-th one. Also, S_i is the amount of samples considered on the i-th node, while v_j and $v_j^{'}$ are the expected and the forged j-th sample, respectively. The forged sample $v_j^{'}$ can be expressed as $v_j^{'} = v_j + \epsilon$, with $\epsilon \in \mathbb{R}$. Though the adversary has no particular restrictions on the choice of ϵ, she has to be careful that $v_j^{'}$ is still valid from a system standpoint.

$$y_A = \frac{\sum_{i=1}^{N} p_i}{N} + \frac{\sum_{i=1}^{L} r_i \cdot ((\sum_{j=1}^{N_i} p_{ij})/N_i)}{L} \qquad (2)$$

Equation 2 measures y_A, i.e. the level of compromise in network availability due to attack A. Basically, y_A indicates how much attack A i) impacts in terms of packet dropping; and ii) results in missed delivery deadlines, thus failing to satisfy possible real-time constraints. More in detail, the first addend reports the impact of A in terms of packet dropping, where N is the amount of discarded packets, and p_i is the weight associated to the i-th one. The second addend provides an indication of how much attack A results in missed deadlines due to delayed packet reception. Specifically, L is the amount of considered recipient nodes, whereas r_i is the weight associated to the i-th one. N_i represents the amount of packets that have missed a deadline on the i-th recipient node. Finally, p_{ij} stands for the weight associated to the deadline missed by the j-th packet expected on the i-th recipient node.

$$z_A = \frac{\sum_{i=1}^{N} p_i}{N} \qquad (3)$$

Finally, Equation 3 measures z_A, i.e. the level of compromise in information confidentiality due to attack A. Basically, z_A indicates how much the adversary has been able to compromise confidentiality, through unauthorized inspection of network packets. We define N as the amount of packets intercepted by the adversary. Then, p_i is the weight associated to the i-th intercepted packet.

$$E_A = w_1 \cdot x_A + w_2 \cdot y_A + w_3 \cdot z_A \qquad (4)$$

Given the three security metrics described above, the overall impact E_A of an attack A can be computed according to Equation 4, where w_1, w_2, and w_3 are the weights associated to service integrity, network availability, and information confidentiality, respectively. Computing such security metrics requires to collect specific information, including the amount and kinds of packets received by recipient nodes, the expiration of real time deadlines, and the occurrence of packet interception by compromised nodes. Thus, application level information alone might not be sufficient to compute security metrics. In other words, we also need information related to the network behavior and the actual communication among sensor nodes. Therefore, unless the considered system model relies on very strong, and possibly unrealistic, assumptions, we believe that network and attack simulation is a valuable and essential approach to gather essential information, and perform the attack ranking process.

6.2. Attack severity evaluation

In this section, we rely on the security metrics defined in Section 6.1, and provide a rank of the attacks considered in Section 5, according to their severity. Since the reception of incorrect data may lead to take incorrect actions, application integrity is typically considered to be more important then network availability in WSN scenarios [2]. For the sake of brevity, in the following we consider only how attack impact on application integrity, i.e. we assume $w_1 = 1$, and $w_2 = w_3 = 0$ in Equation 4.

For each considered attack, we compute the following two impact values. E_{C1} refers only to Cluster *C1*, i.e. it takes into account average values computed by Cluster Head *CH1*. On the other hand, E_S refers to the impact on the system as a whole, i.e. it takes into account average values computed by the Sink node. For both of them, we assume $M = 1$, $r_1 = 1$, and $S_1 = 60$ in Equation 1. Also, we consider i) different injection intervals for the injection attack, i.e. 25, 35, 50, and 70 packets per second; ii) different displacement distances for the misplace attack, i.e. 2, 5, and 10 meters; and, finally, iii) different wormhole intervals for the wormhole attack, i.e. 23.3, 35, and 70 packets per second.

Position	E_{C1}	Attack	Details
#1	57.203	Wormhole	23 packets/s
#2	47.159	Wormhole	35 packets/s
#3	31.655	Wormhole	70 packets/s
#4	19.735	Misplace	10 meters
#5	10.493	Injection	25 packets/s
#6	7.751	Injection	35 packets/s
#7	7.048	Misplace	5 meters
#8	5.297	Injection	50 packets/s
#9	3.504	Injection	70 packets/s
#10	1.332	Misplace	2 meters

Table 1. Attack rank (Cluster $C1$).

Position	E_S	Attack	Details
#1	7.056	Wormhole	23 packets/s
#2	5.906	Wormhole	35 packets/s
#3	4.168	Wormhole	70 packets/s
#4	1.135	Injection	25 packets/s
#5	0.921	Misplace	10 meters
#6	0.833	Injection	35 packets/s
#7	0.564	Injection	50 packets/s
#8	0.499	Misplace	2 meters
#9	0.389	Misplace	5 meters
#10	0.369	Injection	70 packets/s

Table 2. Attack rank (System).

Table 1 and 2 report the attack rank according to the computed value of E_{C1} and E_S, respectively. Since the adversary thwarts Cluster *C1* activities, the impact of each attack is more severe from a cluster standpoint rather than from a whole system point of view. Also, the wormhole attack always displays the most severe impact against service integrity. Of course, the lower the wormhole interval is, the higher the attack impact we observe. Finally, service integrity in Cluster *C1* is more affected by the misplace attack, while the injection attack results to be more effective from a whole system standpoint.

7. Countermeasure evaluation

In this section, we show how it is possible to protect the network by adopting specific security countermeasures. We consider different security solutions, and evaluate both their effectiveness and efficiency through simulation. At the moment, although it relies on quantitative simulation results, the choice and configuration of security solutions is based on heuristic considerations as well as the adoption of best practices. Future works will introduce security metrics aimed at ranking security countermeasures, according to their effectiveness and efficiency.

7.1. Injection attack

In the following, we discuss how to cope with the injection attack, considering an injection interval $I = 35$ ms. A solution against the injection attack consists in *authenticating* report messages, by means of *Message Authentication Code* (*MAC*). So doing, both integrity and authenticity of sent information are assured. Examples of relevant MAC are *HMAC* (e.g. *SHA-1* and *SHA-256*) and *CBC-MAC*. With reference to our application scenario, we can assume that all nodes in Cluster *C1*, including Cluster Head *CH1*, agree on a MAC *H*.

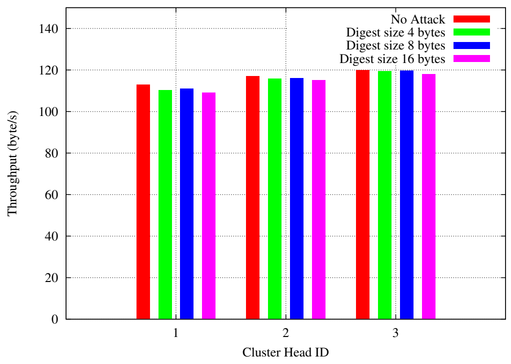

Figure 6. Throughput vs. digest size.

Figure 6 shows the impact of authentication on network performance. Being digests pure additional overhead, they result in a slight throughput decrease on Cluster Head nodes. Also, such a decrease is higher in case a larger digest is used. However, given the nature of the network traffic, and especially the limited data rate, introducing digests has a very low impact on throughput, thus its effect on performance is barely perceptible.

Similarly, Table 3 compares energy consumption in case digests of different sizes are used. Specifically, we

	Energy consumption (mJ)			
	No attack	**Injection attack w/ digest**		
		4B digest	**8B digest**	**16B digest**
Node C1	34.691	34.615	35.110	34.919
Node C2	33.597	33.378	33.704	33.996
Node C3	34.435	34.603	34.828	34.898
Node n5	33.590	33.749	34.054	34.045
Node n8	33.487	33.671	34.832	34.039
Node n10	33.698	33.860	34.034	34.065

Table 3. Energy consumption.

consider two nodes per Cluster, i.e. the Cluster Head node and one sensor node. As we can see, introducing digests results in a slight increase of energy consumption, for both sensor nodes and Cluster Head nodes. This is mainly due to the additional transmission and reception of digests, which obviously increases the radio activity. Besides, as previously discussed for the throughput, also energy consumption is barely affected by message authentication.

Furthermore, report authentication is extremely effective in totally neutralizing the injection attack. In fact, the adversary would be required to either guess the right digest values, or inject a huge amount of fake reports, trying all possible digest values. However, the latter would result in a prohibitive transmission rate, and would be actually more similar to a Denial of Service attack. Having said that, we claim that relying on 4 byte digests is more than sufficient to practically protect the network against the injection attack. Nevertheless, in order to contrast a possible adversary provided with plentiful of computation and transmission resources, it would be possible to rely on larger digests without significantly impacting on network performance and nodes' energy consumption.

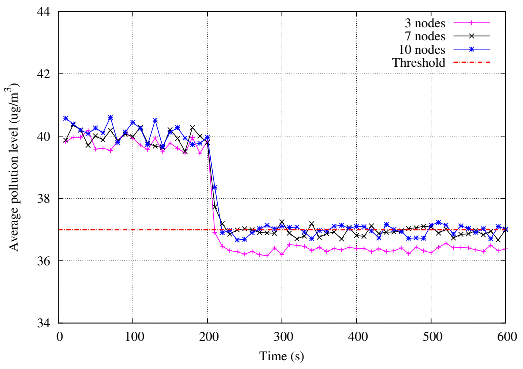

Figure 7. Pollution level (injection).

A different solution aimed at contrasting the injection attack consists in providing *node redundancy*. With reference to our case study, deploying additional sensor nodes in Cluster *C1* should contribute to reduce the attack effectiveness. In fact, in the presence of more sensor nodes, the Cluster Head node can rely on a larger amount of genuine reports, thus limiting the impact of forged reports in the average pollution level computation.

In Figure 7, we compare the effects of the injection attack in the presence of 3, 7, or 10 sensor nodes in Cluster

C1. As we have already shown in Figure 3, in the presence of 3 sensor nodes, the adversary successfully manages to make smokestack *S1* behavior appear as regular. Instead, if more sensor nodes would be part of Cluster *C1*, i.e. 7 or 10, the injection attack would be neutralized. In fact, both in the presence of 7 or 10 sensor nodes, the perceived average pollution level in Cluster *C1* displays values higher than the threshold, thus the injection attack is practically neutralized. However, lines associated to 7 and 10 nodes in Figure 7 are quite close to the threshold, and sometimes even below it. Thus, increasing node redundancy results to be less effective than message authentication.

Also, deploying additional sensor nodes not negligibly increases network activity. In particular, it results in more competition to access the medium, thus increasing transmission contention and packet collisions. Let R be the ratio between the amount of reports successfully received by Cluster Head *CH1* and the amount of lost reports, due to access contention or packet corruption. According to our results, if only 3 sensor nodes are present in Cluster *C1*, the reception ratio R is equal to 2.817. Instead, if Cluster *C1* included 7 or 10 sensor nodes, we would have a reception ratio $R < 1$, namely equal to 0.856 and 0.529, respectively. That is, the amount of lost packets would exceed the amount of packets successfully received by Cluster Head *CH1*. Thus, increasing node redundancy displays a limited effectiveness against the considered injection attack, and severely impacts on network performance.

Our evaluation suggests that, in order to neutralize the considered injection attack, it is better to rely on report message authentication than providing node redundancy, both from an effectiveness and efficiency standpoint.

7.2. Misplace attack

Results presented in Section 5.2 suggest us that the misplacement attack can severely affect report goodness, and effectively alter the computation of average pollution levels. In order to neutralize it, a possible solution consists in providing *physical protection* of sensor nodes, i.e. deploying the network in order to prevent them from being moved from their original position. However, if we refer to results shown in Figure 4, we can claim that it is not necessary to protect the whole area associated to Cluster *C1*. In fact, physical protection is actually required only for sensor nodes close to smokestack *S1*, which are, of course, the most attractive targets for the adversary. For instance, in our specific case, it should be sufficient to physically protect a square area centered at smokestack *S1*, whose side is 30 meters in size. Valid alternative solutions consist in relying on *secure data aggregation* [11][14], or providing *secure localization* in order to promptly detect misplacement of sensor nodes [8][10].

7.3. Wormhole attack

Unlike the injection attack, this attack cannot be contrasted by relying on message authentication. In this case, the legitimate node *n5* is first misplaced, and then compromised to perform the actual wormhole attack. Then, even after being misplaced, *n5* would be able to correctly authenticate its own report messages, before delivering them to Cluster Head *CH1*. Thus, message authentication is not a valid solution against wormhole attack.

However, as we have seen for the injection attack, also the wormhole attack admits node redundancy as a possible solution. That is, the presence of more sensor nodes in Cluster *C1* would make Cluster Head *CH1* rely on a larger amount of genuine reports, thus hopefully limiting the effects on the average pollution level computation.

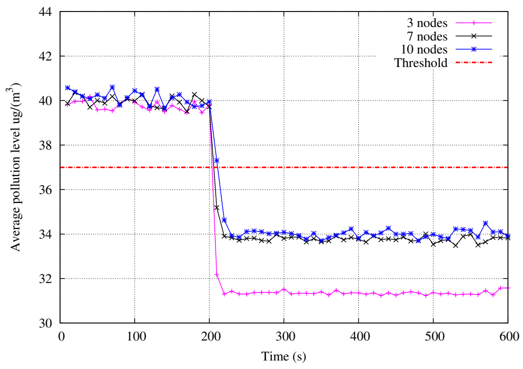

Figure 8. Pollution level (wormhole).

In Figure 8, we compare the effects of the wormhole attack in the presence of 3, 7, or 10 sensor nodes in Cluster *C1*. As we have shown in Figure 5, in the presence of 3 sensor nodes, the adversary manages to make smokestack *S1* behavior appear as regular. However, even in the presence of 7 or 10 sensor nodes, the perceived average pollution level in Cluster *C1* keeps on displaying a value lower than the threshold, and the wormhole attack is not neutralized. Note that the presence of 10 sensor nodes practically results in the same effects shown for 7 sensor nodes only.

Also, as discussed in Section 7.1 for the injection attack, deploying additional sensor nodes noticeably impacts on network activity, by increasing transmission contention and packet collisions. According to our results, if only 3 sensor nodes are present in Cluster *C1*, the reception ratio R is equal to 3.559. However, in the presence of 7 or 10 sensor nodes, we have a reception ratio $R < 1$, namely equal to 0.966 and 0.572, respectively. That is, the amount of lost packets in Cluster *C1* exceeds the amount of packets successfully received by Cluster Head *CH1*.

Thanks to our quantitative evaluation, we can claim that increasing node redundancy displays a severe impact on network performance, and, even worse, results to be ineffective in neutralizing the considered attack. Morever, deploying more than 10 sensor nodes in Cluster *C1* is not likely to provide a significant contribution in better contrasting the attack. Also, it would further worsen network performance, by reducing reception ratio in Cluster *C1*. Therefore, node redundancy should not be considered as a valid solution against wormhole attack. Instead, it would be better to rely on alternative solutions, as physical protection of sensor nodes or secure localization [8][10].

8. Conclusion

In this paper, we have presented our simulative approach to attack impact analysis in WSNs. Our approach allows for evaluating the effects of attacks, ranking them according to their severity, and provides valuable insights on the attack impact since during the design phase, thus helping to devise security priorities and select appropriate countermeasures. In order to support our points, we have considered a pollution monitoring application as a case study. We have evaluated the impact of cyber-physical attacks, discussed the attack ranking process, and analyzed different countermeasures. We believe our approach can be of great help during the risk assessment process, and hope it will be considered as part of best practices for designing secure WSNs. Future works will introduce security metrics aimed at ranking security countermeasures, according to their effectiveness and efficiency.

Acknowledgment

This work has been supported by EU FP7 Integrated Project PLANET (Grant agreement no. FP7-257649) and the TENACE PRIN Project (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research. We also thank Alessandro Pischedda for his valuable help.

References

[1] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang and S. Sastry. Attacks Against Process Control Systems: Risk Assessment, Detection, and Response. In *The 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 355–366, New York, NY, USA, 2011. ACM.

[2] A. A. Cárdenas, T. Roosta, and S. Sastry. Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems. *Ad Hoc Networks*, 7(8):1434–1447, November 2009.

[3] A. Becher, E. Becher, Z. Benenson and M. Dornseif. Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks. In *The 3rd International Conference on Security in Pervasive Computing (SPC 2006)*, pages 104–118, 2006.

[4] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In *The First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, May 2003.

[5] G. Dini and M. Tiloca. ASF: an Attack Simulation Framework for wireless sensor networks. In *The 8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2012)*, pages 203–210, October 2012.

[6] G. Padmavathi and D. Shanmugapriya. A Survey of Attacks, Security Mechanisms and Challenges in Wireless Sensor Networks. *International Journal of Computer Science and Information Security*, 4(1&2):117–125, August 2009.

[7] G. Stoneburner, A. Goguen and A. Feringa. Risk Management Guide for Information Technology Systems - Recommendations of the National Institute of Standards and Technology. Technical report, National Institute of Standards and Technologies, July 2002.

[8] L. Lazos and R. Poovendran. SeRLoc: Robust localization for wireless sensor networks. *ACM Transactions on Sensor Networks*, 1(1):73–100, August 2005.

[9] R. Daidone, G. Dini and M. Tiloca. On experimentally evaluating the impact of security on IEEE 802.15.4 networks. In *The 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS 2011)*, pages 1–6, June 2011.

[10] S. Capkun and J.-P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):221–232, February 2006.

[11] S. Ozdemir and Y. Xiao. Polynomial Regression Based Secure Data Aggregation for Wireless Sensor Networks. In *The 2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*, pages 1–5, December 2011.

[12] T. Bonaci, L. Bushnell and R. Poovendran. Node capture attacks in wireless sensor networks: A system theoretic approach. In *The 49th IEEE Conference on Decision and Control (CDC 2010)*, pages 6765–6772, December 2010.

[13] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *The 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 171–180, New York, NY, USA, 2003. ACM.

[14] V. Kumar and S. Madria. Secure Hierarchical Data Aggregation in Wireless Sensor Networks: Performance Evaluation and Analysis. In *The 13th IEEE International Conference on Mobile Data Management (MDM 2012)*, pages 196–201, July 2012.

[15] V.C. Gungor and G.P. Hancke. Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches. *IEEE Transactions on Industrial Electronics*, 56(10):4258–4265, 2009.

[16] Y. Hu, A. Perrig and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In *IEEE INFOCOM 2003*, April 2003.

[17] Y.-L. Huang, A. A. Cárdenas, S. Amin, Z.-S. Lin, H.-Yi Tsai and S. Sastry. Understanding the physical and economic consequences of attacks on control systems. *International Journal of Critical Infrastructure Protection*, 2(3):73–83, 2009.

[18] Y. Mo, T. H.-J. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig and B. Sinopoli. Cyber-Physical Security of a Smart Grid Infrastructure. *Proceedings of the IEEE*, 100(1):195–209, January 2012.

[19] Y.-T. Wang and R. Bagrodia. SenSec: A Scalable and Accurate Framework for Wireless Sensor Network Security Evaluation. In *The 31st International Conference on Distributed Computing Systems Workshops (ICDCSW 2011)*, pages 230–239, June 2011.

[20] Y. Xu, G. Chen, J. Ford and F. Makedon. Detecting Wormhole Attacks in Wireless Sensor Networks. In Eric Goetz and Sujeet Shenoi, editor, *Critical Infrastructure Protection, Post-Proceedings of the First Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection*, volume 253 of *IFIP*, pages 267–279. Springer, March 2007.