Wireless Pers Commun (2015) 81:1359–1376
DOI 10.1007/s11277-015-2478-z

Multi-Metrics Approach for Security, Privacy and Dependability in Embedded Systems

Iñaki Garitano[2] · Seraj Fayyad[1,2] · Josef Noll[1,2]

Published online: 13 March 2015

Abstract Embedded Systems have become highly interconnected devices, being the key elements of the Internet of Things. Their main function is to capture, store, manipulate and access data of a sensitive nature. Moreover, being connected to Internet, expose them to all kind of attacks, which could cause serious consequences. Traditionally, during the design process, security, privacy and dependability (SPD) have been set aside, including them as an add-on feature. This paper provides a methodology together with a Multi-Metrics approach to evaluate the system SPD level during both the design and running processes. The simplicity, based on a single process during the whole system evaluation, and scalability, simple and complex systems are evaluated equally, are the main advantages. The applicability of the presented methodology is demonstrated by the evaluation of a smart vehicle use case.

Keywords Internet of Things · Embedded Systems · Security · Privacy · Dependability · Multi-Metrics · Sensor systems

1 Introduction

Our society is build and driven by Embedded Systems (ESs). Composed of different components, ESs range from low-end systems, such as smart cards, to high-end systems, like routers and smart phones. Furthermore, ESs constitute one of the key elements of the

✉ Iñaki Garitano
igaritano@garitano.org; igaritano@unik.no

Seraj Fayyad
seraj@unik.no

Josef Noll
josef@unik.no

[1] University of Oslo, Oslo, Norway

[2] UNIK, Kjeller, Norway

Internet of Things [1]. The technological progress produced several effects, such as the power and performance boost of ESs. Hence, their capabilities and services have raised, and in consequence, their usage has been substantially increased.

Together with the evolution of performance, energy consumption and size, ESs jump from isolated environments to interconnected domains. Although the evolution of connectivity enlarges the number of possible services, at the same time it increases the *attackability* of this kind of systems.

Embedded Systems are used for multiple purposes, mainly to capture, store and control data of sensitive nature, e. g. car systems. Attackers could have different goals to compromise ESs, from gathering sensitive data, thus compromising their privacy, to disrupt the service by a Deny of Service (DoS) attack, exploiting their security and dependability. The consequences of a malicious and a successful attack could cause physical and economical losses, and thus it is important to keep them as secure, privacy-aware and dependable as needed in a given situation.

Traditionally, the design process of ESs has been focused on reducing costs, size and the energy consumption while increasing their performance and power. security, privacy and dependability (SPD) have been applied as add-on instead of built-in features. Furthermore, rather than as a group, these properties have been analysed and implemented individually which decreases their efficiency. Although this approach reduces the design and development costs, it sacrifices the quality of applied measures converting ESs more vulnerable. In order to create secure, privacy-aware and dependable ESs, the design process must tackle SPD from the beginning, which will provide robust systems.

Within this paper, a functional SPD level evaluation methodology is presented and its applicability validated by implementing it in a real use case. The core of the methodology resides in the Multi-Metrics SPD evaluation, which provides a practical and simple solution for SPD implementation during not only the design, but the whole lifetime of ESs. The presented concepts and results are developed through the European activity SHIELD. The nSHIELD project [2] looks at the applicability of the envisaged approach in different domains. This paper focusses on privacy in a social setting of lending a vehicle.

The rest of the paper is structured as follows: Sect. 2 provides an overview of related work on security, privacy and dependability; Sect. 3 presents the ESs SPD level methodology; Sect. 4 introduces the smart vehicle use case and explains its main features; Sect. 5 defines the metrics selection and definition process, and describes the five metrics using along the work; Sect. 6 introduces the Multi-Metrics approach and shows its applicability by analysing the privacy SPD level, SPD_P, of a smart vehicle sub-system; Sect. 7 evaluates the results obtained in the previous section and finally, Sect. 8 provides a summary of the key contributions of the research.

2 Related Work

This section describes some of the works related with security, privacy and dependability metrics and their measurement. Furthermore, the main aspects of each work are described and compared with the solution presented within this paper. The reviewed works are classified into three main SPD aspects.

There are two types of security, privacy and dependability metrics: system-based and attacker-based measurements. The first ones are called as system-centric approaches, while the last ones are known as attacker-centric methods [3]. Those approaches focused in

attackers, assume attacker capabilities, resources and behaviour, while system-centric solutions dismiss assumptions, and rather concentrate in system components and capabilities. Previous research on security measurement has been mainly concentrated on attacker-centric approaches [4, 5], even if there are some system-centric methods [3].

From the Embedded Systems development point of view, the focus on attacker capabilities and behaviour is not helpful, as it leaves out the system-centric design and configuration aspects.

2.1 Security Metrics

One of the well-represented studied areas deals with the security of different types of system, considering security, privacy and dependability metrics individually.

Prior research on security metrics has been mainly focused on software security. Furthermore, software security has been analysed from different layer perspective: code, function and system levels.

Code level metrics analyse code bugs, but without weighting them, i.e., all bugs have the same importance. Intuitively, by lowering the amount of code bugs, they increase the overall system security.

A wide group of researches have focussed on detection of bugs at code level [6–9]. One of the main problems of this approach is that giving the same importance to all bugs, it does not represent which bugs are easier to exploit.

System level security metrics rely on organizations and websites that publish discovered vulnerabilities in various systems. Some of those organizations and websites are US-CERT [10], NIST [11], MITRE [12] and SecurityFocus [13]. Browne et al. [14] create a mathematical model to measure the frequency at which incidents related to a specific vulnerability are reported to the US-CERT. In addition, Alves-Foss et al.[15] calculate the system vulnerability index using system characteristics, potentially neglectful and malevolent acts factors, to use it as a measure of computer system vulnerability. Furthermore, Beattie et al. [16] propose a model which finds the appropriate time for applying security patches to a system for optimal uptime. Beattie's work is not just focusing on security, but also addresses dependability. Brocklehurst et al. [17] include the effort that an attacker needs and the obtainer reward factors to measure the operational security of a system. The main restriction of system level security metrics is that they ignore specific system configurations which allow or prevent that a particular vulnerability might be exploited.

Function level metrics weight the bugs giving them a different importance, thus provide a more specific approach. Moreover, function level metrics analyse vulnerabilities taking into account the system conditions, which is represented by the *attackability* concept. Since the attack surface metric was firstly introduced by Michael Howard [18], it has been applied in different domains by multiple works [19–25]. Defined as the *attack opportunity* or *attackability* of a system, or its exposure to attack [18, 20], attack surface is a relative metric that strikes at the design level of a system. Howard et al. [20] propose the attack surface metric for determining whether one version of a system is more secure than another with respect to a fixed set of dimensions. Their work evaluates the attack surface metric of five different versions of Windows operating system. To do so, they define and use five different elements to evaluate the attack surface level; *Target, Enabler, Channel, Protocol and Access rights*. After giving a specific weight to each element, which reflects the repercussion of each of them, all the elements are computed with a function resulting in attack surface level. The main advantage of this method is that dividing the metric into small elements helps to simplify the approach. However, the function for computing all the

elements together have to be specified for each system, which, even for a simple system, can be extremely difficult. Additionally, this work relies on the history of attacks on a system, which prevents it for applying in a systematic form.

Different variations of an attack surface metric have been introduced. Manadhata and Wing [22] modified the attack surface metric introduced by Howard et al. [20] in order to categorize the system resources into different *attack classes*. The main idea behind classifying system resources is based on the notion that some of them are more likely to be attacked than others. After identifying and classifying all attackable system resources, they presented, measured and compared the attack surface of two Linux distributions, Debian [26] and Red Hat [27]. Following the same approach, Manadhata et al. analyse and compare the attack surface of two IMAP servers [23, 28, 29] and two FTP Daemons [23, 29, 30].

One of the essential ideas behind attack surface metric is that is important to remove unnecessary features, and offer those characteristics as reconfigurability or *composability* options. In this sense, and considering the flexibility to add/remove specific component from some operative system kernels, Kurmus et al. [21, 31, 32] suggest a dynamic analysis of system features to compose an OS kernel just with necessary features. The main problem of this approach is that the characterization of a operating system and its applications in a determinate moment helps to reduce its attack surface but it prevents to expand with more features later on. Stuckman and Purtilo [24] continue with the *composability* idea by proposing a method to evaluate the attack surface impact of a configuration or environmental change, which helps system administrators to tune their applications to optimize security. The reduction of the attack surface on permission-based software has been also analysed by Bartel et al. [19]. Their work has been fully implemented on Android, a permission based platform for mobile devices.

Embedded Systems are a combination of hardware and software elements working together to offer different services. One of the main problems of software security metrics is that they just focus on a single software package or operating system. They do not consider the security of a combination of software packages. Thus, they do not include the interrelation between different systems. From the ESs perspective, is necessary to analyse not only the security of a single element, but the combination of multiple components. Hence, the Multi-Metrics approach suggested within this work, first evaluates the SPD level of single components to finally join all of them and obtain an overall SPD_{System} level.

2.2 Privacy Metrics

Privacy metrics research is not as extensive as security metrics research presented in the previous section. Additionally, most of the works which propose privacy measurements focus on user location privacy.

Krumm [33] proposes two location privacy metrics emphasizing the importance of finding a single quantifier for location privacy. Shokri et al. [34] first analyse various location-privacy protection mechanisms and introduce three different metrics used for measuring location privacy; uncertainty-based, error-based and k-anonymity metrics. After comparing those metrics their work states that metrics such as entropy and k-anonymity are inadequate for measuring location privacy. Additionally, Shokri et al. propose a formal framework for the analysis and the systematic comparison of location-privacy protection mechanisms. The framework proposed by their work is an attack-centric method which considers the prior information that could be available to attackers, and various attacks that can be performed. The location-privacy framework considers elements such as mobile

users, possible traces of a specific users, location-privacy preserving mechanism, observable traces for the attacker, inference attacks from the adversary and evaluation.

The work of Ma et al. [35] is an example of privacy research related to vehicular communication systems. Their work uses the entropy as the metric to measure the location privacy level.

Within this paper the privacy level SPD_P, for three scenarios of a smart vehicle is calculated. In each scenario, the privacy level is measured in term of user location privacy. However, instead of analysing the privacy by its own, the resulting level is associated with security and dependability levels. Along Sect. 7, the result of combining privacy and dependability will show the necessary commitment to reach an overall SPD_{System} level.

2.3 Dependability Metrics

Dependability level is measured by the combination of availability, reliability, safety, integrity and maintainability. Even if all dependability elements are measurable, just availability and reliability are quantifiable. Thus, traditional dependability metrics are related with those two elements; Mean Time Between Failures (MTBF), Mean Time To Repair (MTTR), Mean Time To Failure (MTTF), Mean Time To Unsafe Failure (MTTUF) are some of them. Prior research analyses dependability either during the design process or under operational conditions.

Jatain et al. [36] compare software reliability measurement models which use design phase dependability metrics. Some of the metrics used by the analysed models are the average expected lifetime, elapsed time between failures, number of failures, mean time between failures and fault rate. The main problem of comparing models resides in the fact that none of them can be adjusted to a specific system. In other words, all models base their analysis on a predefined set of metrics, which can not be extended to include aspects or configurations of a specific system. Therefore, their approach is less suitable for comparing different configuration options of a specific system.

Henkel et al. [37] present a multi-layer dependability design flow. As it is defined within the work, the multi-layer term is use to form and/or adapt together at least two hardware and/or software abstraction layers, during the system design or runtime. The main advantage of the multi-layer approach resides in the reduction of error propagation. Each layer is evaluated independently and the result is combined with other layers to end up with an overall dependability value. The errors are addressed within each layer which prevents the extension of their effects towards other layers. Our methodology proposed in Sect. 3 is also based on the evaluation of the SPD level of different layers. The Multi-Metrics approach eases the understanding of the compliance of system components or sub-systems, providing a simple view of what to solve to end up with an appropriate overall SPD_{System} level.

Weiner et al. [38] use different metrics to evaluate the dependability of the proposed wireless system architecture. The metrics they propose are used to measure the dependability of wireless sensors and actuators for a control unit. Signal propagation delay, probability of failure and worst case latency are assessed using metrics. These metrics are used within the proposed model to evaluate the worst-case latency and compare it with the critical latency. The result is used to evaluate if actual wireless communication systems are appropriate enough for their system requirements. While Weiner et al. focussed on dependability, our approach evaluates the SDP level of different configurations and shows which of them are not suitable to satisfy the established SPD_{Goal}.

Prior research analysed security, privacy and dependability measurements individually. Even if some of them combine security and privacy or security and dependability, none of them combines all of them. Considering SPD independently in designing fragile systems could result in a highly secure system, but the system might be highly vulnerable due to dependabilities. In most cases a specific security, privacy or dependability level requires a compromise between one or the other, implying a balance between all three of them. The next section describes a methodology to combine and evaluate different metrics by the usage of the Multi-Metrics approach.

3 Methodology for Security, Privacy and Dependability

This section describes the methodology to measure the security, privacy and dependability (SPD) level of a system. The objective is to achieve an overall system SPD level, SPD_{System}. The main advantage of this methodology is that it provides a simple mechanism to measure and evaluate the system security, privacy and dependability levels.

SPD_{System} is a triplet, composed of individual security, privacy and dependability levels (s, p, d). Each of the levels is represented by a range between 0 and 100, i.e. the higher the number, the higher the security, privacy and dependability level. However, in order to end up with SPD_{System}, during the whole process, the criticality is evaluated. Criticality is again a triplet (Cs, Cp, Cd), defined as the complement of SPD, and expressed as $(Cs, Cp, Cd) = (100, 100, 100) - (s, p, d)$.

As shown in Fig. 1, a system is composed of multiple sub-systems which at the same time consist of various components. Additionally, a system is used in different scenarios, and for each scenario the configuration of sub-systems and components could be different.

Every scenario in which the system will run have some SPD requirements, SPD_{Goal}. Thus, the system will have as many SPD_{Goal} as scenarios. In addition, for each scenario, the system can be configured differently, which derives in a distinct SPD level for each configuration in every scenario. Therefore, the introduced methodology evaluates all configurations on every scenario, and it ends up by comparing and selecting the configuration that best satisfies the established SPD_{Goal}.

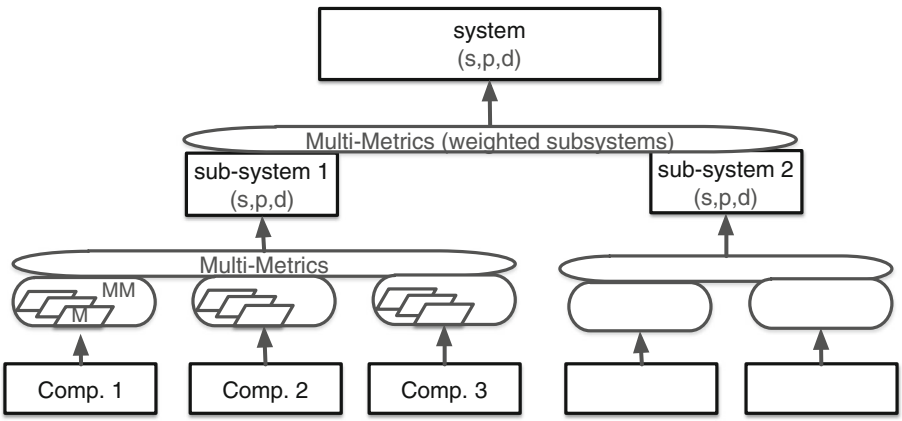


Fig. 1 System level Multi-Metrics (MM), with M indicating a Metrics analysis

The methodology described within this section, is composed using the evaluation of components, sub-systems and the system. As shown in Fig. 1, the methodology starts by evaluating each component, then the combination of components that make up a sub-system, and finally the set of sub-systems that forms the system. Between every system division, the Multi-Metrics approach is applied. Applicability of the Multi-Metrics will be further explained within Sect. 6.

The metrics are the objects or entities used to measure the criticality of components. As presented in Fig. 2, the criticality of a component for a given configuration is evaluated through one or more metrics. The result of the metrics is again joined by using Multi-Metrics, which provides the overall component measurement. The selection and/or definition of metrics, applied for a specific use case, is further explained in Sect. 5.

Each component, each sub-system, and the SPD level of the system is computed for all possible configurations supporting a given scenario. Security, privacy and dependability of the system are evaluated individually. However, whenever the obtained SPD level have to be compared with the scenario SPD_{Goal}, all SPD elements are taken into account, and a visual representation is used for the SPD_{System} compliance.

The main purpose of the visual SPD level representation consists of simplifying the comparison between the SPD level of each configuration and the SPD_{Goal} for a given scenario. The rest of the work is focused just on privacy level in order to simplify the example. Each scenario has a predefined SPD_{Goal}, as shown in Table 1, and the evaluation of each configuration results in a specific SPD level.

In order to simplify the selection of the most suitable configuration, every element of SPD level is substituted by a *green*, *yellow* or *red* circle. The colour is selected according to the numeric difference between SPD level and SPD_{Goal}, following the next criteria:

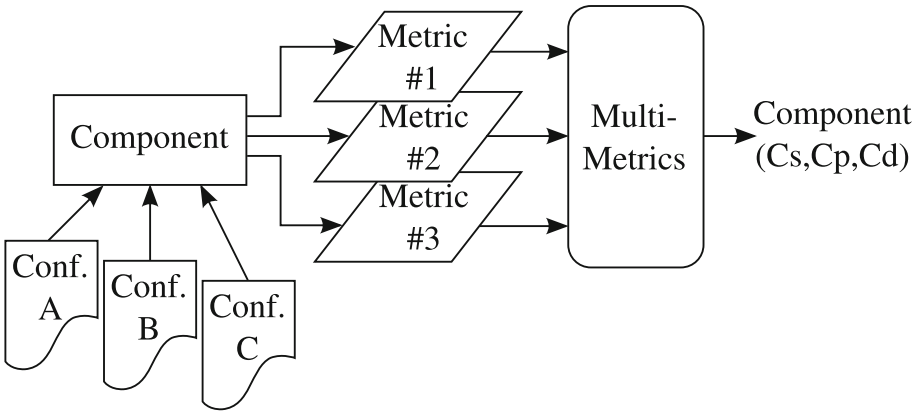


Fig. 2 Component level Multi-Metrics

Table 1 SPD_{Goal} and SPD level. (Color table online)

		SPD_{Goal}	SPD level	
Scenario 1	Conf. A	(s, 80, d)	(s, 100, d)	(s, ●, d)
	Conf. B		(s, 80, d)	(s, ●, d)
	Conf. C		(s, 80, d)	(s, ●, d)

- $|\text{SPD}_{Goal} - \text{SPD level}| = \leq 10$, green ●.
- $|\text{SPD}_{Goal} - \text{SPD level}| = > 10, \leq 20$, yellow ●.
- $|\text{SPD}_{Goal} - \text{SPD level}| = > 20$, red ●.

Table 1 shows the colour representation of the privacy level for each system configuration.

The last step, after following the same approach with security and dependability measures, is to make a commitment and select the most convenient configuration for a given scenario. As result, the SPD_{System} will be established.

Within this chapter, the methodology for evaluating and selecting the best SPD level according to the SPD_{Goal} has been presented. Its main advantage consists in the simplicity of evaluating and selecting the most appropriate configuration for a given scenario. The Multi-Metrics approach reduces the complexity of the evaluation process; the visual representation of SPD simplifies the selection process.

4 Use Case Smart Vehicle

This section describes the smart vehicle use case which is used, along the paper, to show the applicability of the presented methodology. The use case includes three different scenarios and nine possible system configurations. The main purpose is to analyse which of the configuration best fits the envisioned SPD_{Goal} of each scenario and evaluate the SPD level in which the system will run.

4.1 System Description

Our use case envisage the privacy aspects of motorbike (MC) riding. A young driver is allowed using the MC given that he is not speeding ($v \leq 80\,km/h$). His parents ensure him full privacy while keeping within the agreed speed limit (scenario 1), but will be informed if he exceeds the speed limit (scenario 2). In scenario 2, the parents will receive a SMS about the speed and the location of the vehicle. We also add an emergency scenario in case of accidents (scenario 3), where an SMS alert is sent to both parents and emergency services, and where relevant information from the MC is provides to a Backend system.

The Smart vehicle, shown in Fig. 3, consists of three different sub-systems in order to fulfill the requirements from the use case.

- A Backend (BE) system, building the interface from the MC to the end-user,
- An Embedded System (ES), mounted on the vehicle, to monitor the conditions of the MC, and
- The ES—BE communication, being a mobile link between the ES and BE.

4.2 ES—BE Communication Sub-System

This sub-system connects the ES with the BE. Communication is either performed through GPRS or through SMS in case of notices to parents or emergency units. The analysis performed in this paper focusses on just the communication subsystem to ease the understanding of the overall methodology.

The overall system may operate in 9 configurations (A–I), addressing grades of privacy protection:

- Conf. A: The ES does not send any SMS; GPRS data are encrypted with 128 bits key. The ES accepts remote configuration from the BE.

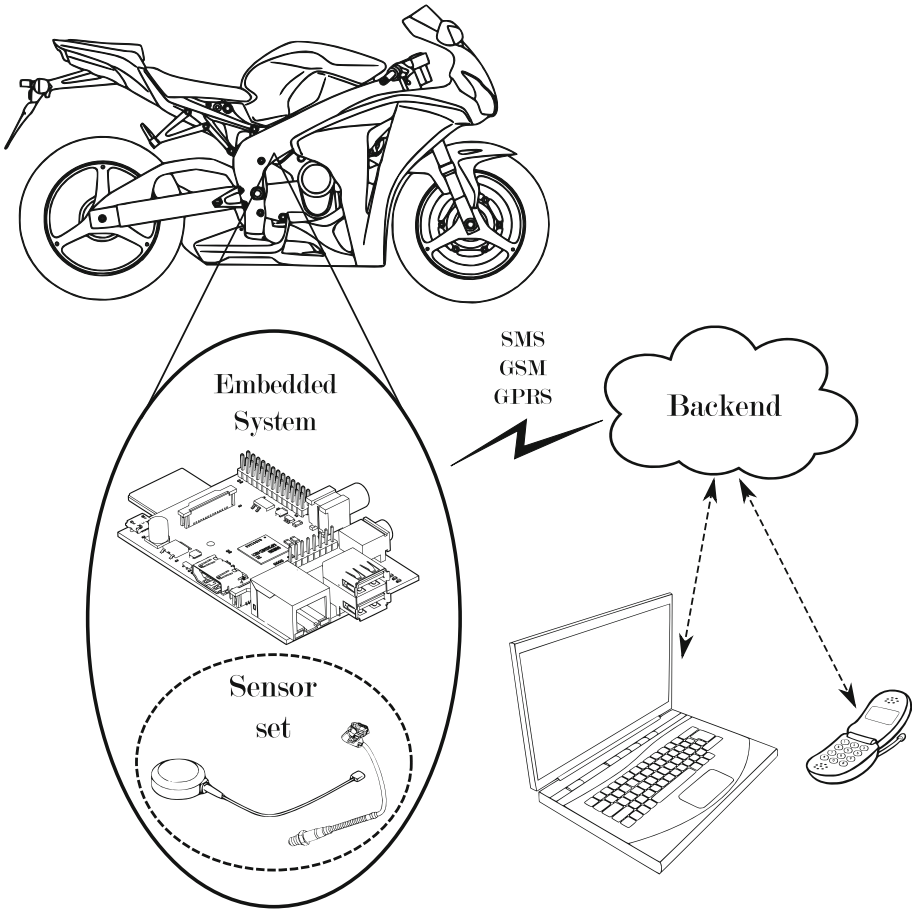


Fig. 3 Smart vehicle topology, consisting of Embedded System (ES) with sensors, communication and Backend System (BE)

- Conf. B: Same as above, except ES sends a keep alive message to the BE every 120 s.
- Conf. C: Same as above, except BE sends messages to the ES and the last one replies every 60 s.
- Conf. D: The ES sends an SMS to parents; GPRS data to the BE are encrypted with 64 bits key. ES accepts remote configuration from the BE.
- Conf. E: Same as above, except ES sends location and speed information to the BE every 10 s.
- Conf. F: Same as above, except BE sends messages to the ES and the last one replies with location and speed information every 5 s.
- Conf. G: ES sends one SMS to parents, another to emergency services. Unencrypted data about the status of the MC are sent from the ES to the BE. ES accepts remote configuration from BE.
- Conf. H: Same as above, except ES sends location and speed information to the BE every 2 s.
- Conf. I: Same as above, except BE sends messages to the ES and the last one replies with location and speed information every 0.5 s.

Having introduced the use case, the architecture and the configuration of the system in this section, the following section will select the metrics to evaluate the system and establish the SPD level.

5 Metrics Definition and Selection

This section describes the selection and definition of SPD metrics. Five metrics are used to evaluate the SPD_{Goal} level of the smart vehicle, being presented in Sect. 4.

The SPD level of the components that make up an Embedded System can be measured by multiple metrics. Definition and selection of the necessary metrics requires expertise in the field, and should be performed by a system engineer. One of the ideas behind this work resides in the creation and maintenance of a common metric database. The main benefit consists of reusing the metrics used to measure the same or equivalent component SPD level, or even use existing SPD values for subsystems/components with a given configuration. To the best of our knowledge, there is no metric database available, giving us the task of defining relevant metrics.

5.1 Metrics Definition

The proposed metric definition methodology is composed of four phases: parameters identification, parameters weighting, component-metric integration and metric running. Our smart vehicle system is evaluated through five metrics, following the four phases shown in Fig. 4.

The Parameter Identification phase involves the component analysis which will be measured by the metric itself. Based on the component purpose and its characteristics, the identified parameters will be classified and transferred to the next phase.

The next step, Parameter Weighting, evaluates the repercussion of the possible values of each parameter on the component SPD level. This step needs to be done by an expert in the field, since it requires a good knowledge not only about the system itself, but also in the security, privacy and dependability domains. The impact of each parameter value for a given SPD aspect is defined by its weight. Notice that the weight will be different for each SPD aspect.

The next phase, the Component-Metric integration, consists of identifying the possible values that a parameter could have based on all the configurations in which a system will run.

After all the parameters, the weight of their values within a component and the possible configuration values have been identified, the last step is to run the metric. In this phase parameter values specified by each of the configuration files are evaluated through the respective metric and provide the criticality level of the component. This step is repeated for every SPD aspect.

5.2 Use Case Metrics

In this work the Multi-Metrics approach has been applied on the Embedded System (ES) and the Backend (BE) communication sub-system. The following five metrics have been

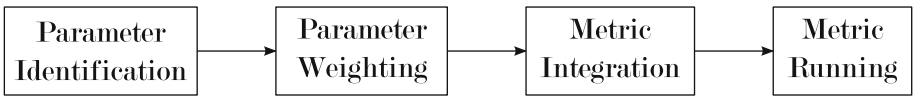


Fig. 4 Metric definition and run phases

defined to measure the criticality level of the four components which make up the evaluated sub-system:

- Ports metric
- Communication channel metric
- SMS message rate metric
- GPRS message rate metric
- Encryption metric

Within the following subsections an overall description of each metric is provided. The port metric is described in detail to explain the four metric definition phases.

5.2.1 Ports Metric

The port metric evaluates the impact of the port component within the ES—BE communication sub-system. Next, the steps followed for its definition are described. Notice that all presented values refer just to privacy evaluation.

1. *Parameter Identification* the analysis of the port component and the ES—BE communication sub-system ends up with the identification of four protocols or port classes: SSH, SNMP, SNMP traps and SMS.
2. *Parameter Weighting* the formulation of the weight, by an expert in a field, provide the values shown in Table 2.
3. *Component-Metric Integration* the identification of all possible port values for a given system configuration provides a considerable amount of options.
4. *Metric Running* Table 3 shows the values obtained from the evaluation of configuration B within the scenario 1.

5.2.2 Communication Channel Metric

The communication channel metric measures the SPD level offered by the two possible channels used by the ES—BE communication sub-system: GPRS data transmission and SMS message. Table 4 shows the parameters of communication channel metric and their weights.

5.2.3 GPRS Message Rate Metric

The GPRS message rate metric evaluates the period in which data are transmitted over the GPRS channel. This metric takes into account the amount of information that could be jeopardized in case the communication channel is compromised. Table 5 shows the parameters of GPRS Message Rate metric and their weights.

5.2.4 SMS Message Rate Metric

The SMS message rate metric evaluates the amount of transmitted messages. This metric takes into account the amount of information that could be jeopardized in case the communication channel is compromised. Table 6 shows the parameters of SMS message rate metric and their weights.

Table 2 Ports metric parameters weights

Parameter	Cp	Characteristitics
SNMP (UDP) 161 in the ES	40	Usage: BE checks if ES is alive.
		Handled data: IP addresses, ports
SNMP trap (UDP) 162 in the BE	60	Usage: ES sends to BE keep alive message
		Handled data: IP addresses, ports
		Usage: ES sends to BE observation data
		Handled data: speed, location.
		Usage: ES sends to BE alert data in accident case
		Handled data: alert, location
SSH (TCP) 23 in the ES	30	Usage: BE sends ES configuration
		Handled data: ES configuration
SMS	80	Usage: ES sends vehicle owner observation data
		Handled data: location and speed of the vehicle
		Usage: ES sends vehicle owner alert in accident case
		Handled data: alert, location, sender phone number, receiver phone number

Table 3 Ports metric integration

Configuration	Metric parameters	Cp
Conf. A	SSH	30
Conf. B	SSH + SNMP trap	61
Conf. C	SSH + SNMP	41

Table 4 Communication channel metric

Parameter	GPRS with GEA/3	SMS over GSM with A5/1
Cp	20	40

5.2.5 Encryption Metric

The encryption metric measures the needed strength to decrypt data, provided by the encryption component. It focuses in the used key length to encrypt the transmitted data. Table 7 shows the parameters of Encryption metric and their weights.

This section introduces the necessary steps to define the metrics used for components criticality evaluation. Furthermore, five metrics used to evaluate the ES—BE communication sub-system are presented. The result of evaluating the four components is further explained within the following section.

6 Multi-Metrics Approach and Operation

This chapter describes the Multi-Metrics (MM) approach and shows its applicability on the sub-system described in the previous section.

Table 5 GPRS message rate metric

Parameter (s)	0.5	1	2	5	10	20	60	120	∞
Cp	80	60	45	30	20	15	10	5	0

Table 6 SMS message rate metric

Parameter	2 messages	1 message	0 message
Cp	10	5	0

Table 7 Encryption Metric

Parameter	No encryption	Key 64 bits	Key 128 bits	Not applicable
Cp	88	10	5	0

Multi-Metrics is the core process of the overall methodology. During the SPD evaluation of an entire system, Multi-Metrics is used in repeated occasions to evaluate the SPD level step by step and end up with the overall SPD_{System}.

Multi-Metrics is a simple process which evaluates the repercussion of each metric, component or sub-system, based on its importance within the system. Its main advantage resides in its simplicity, which is based on the combination of two parameters: the criticality level for a given system configuration and its importance. The criticality level varies between a metric, component or sub-system, depending on the level in which Multi-Metrics is applied. Furthermore, the usage of the same operator along the whole SPD_{System} evaluation simplifies the methodology by making it more understandable for people not being experts in the field. The output of Multi-Metrics is a single number which shows the criticality level of the components and subsystems, and is easily translated into a SPD level.

The importance or significance of a specific metric, component or sub-system, within the system SPD level evaluation, is given by its weight. The weight, is a value within a range of 0 and 100, i.e. the higher the number, the larger is the significance of the evaluated element. The definition of the weight of an element depends on the role or function it performs, and is established by the system designer or an expert in the field.

The Multi-Metrics approach is based on two parameters: the actual criticality x_i and the weight w_i.

The criticality C is accomplished by the root mean square weighted data (RMSWD) formula.

$$C = \sqrt{\sum_i \left(\frac{x_i^2 w_i}{\sum_i^n w_i} \right)} \tag{1}$$

There are three possible criticality level outcomes, being (1) component criticality, after evaluating the suitable metrics, (2) sub-system criticality, from the evaluation of components or (3) system criticality, after performing the Multi-Metrics operation on sub-systems. The actual criticality x_i is the result of (1) the metric for a component evaluation, (2) the component evaluation, obtained by a previous RMSWD, for a sub-system evaluation, or (3) the sub-system evaluation, obtained by a previous RMSWD, for a system evaluation.

All this values are for a given configuration in a specific scenario.

6.1 Communication Subsystem Evaluation

This section performs the evaluation of the ES—BE communication subsystem, and thus demonstrates the the applicability of the methodology, using the privacy evaluation as an example.

The ES and Back end communication sub-system is composed of four components, listed below together with their weight (w) within the sub-system:

1. Port (C1), $w = 40$
2. Channel (C2), $w = 20$
3. Data transmitter (C3), $w = 35$
4. Encryption (C4), $w = 60$

At the same time, the evaluation of those four components is performed though five metrics, described previously in Sect. 5.2. These are the five metrics together with their weigh (w) for the component they evaluate:

1. Port (M1), $w = 100$
2. Communication channel (M2), $w = 100$
3. GPRS message rate (M3), $w = 80$
4. SMS message rate (M4), $w = 20$
5. Encryption (M5), $w = 100$

The SPD_{Goal} for privacy SPD_P is presented in Table 8 and refers solely to the privacy goal of the given scenario. Furthermore, as the evaluation of the sub-system is just considering the privacy p, the security s and dependability d values are not shown.

The results obtained from the evaluation of the privacy for the four components by the five metric values and the sub-system are showed in Table 8. Components C1, C2 and C4 refer each to just one metric, while component C3 has been evaluated by two metrics, M3 and M4. Since the ES—BE communication subsystem is composed of four components, the evaluation of its criticality includes all of them.

Table 8 Multi-Metrics evaluation of the sub-system. (Color table online)

	Criticality					SPD_P			
	C1	C2	C3	C4	Sub-Sys.		Scen. 1	Scen. 2	Scen. 3
SPD_{Goal}							(s, 80, d)	(s, 50, d)	(s, 5, d)
Multi-Metrics elements	M1	M2	M3 ∩ M4	M5	C1 . . . ∩ . . . C4				
Conf. A	30	20	0	5	17	83	●	●	●
Conf. B	61	20	4	5	32	68	●	●	●
Conf. C	41	20	9	5	23	77	●	●	●
Conf. D	82	41	2	10	45	55	●	●	●
Conf. E	82	41	18	10	45	55	●	●	●
Conf. F	83	41	27	10	47	53	●	●	●
Conf. G	82	42	4	88	70	30	●	●	●
Conf. H	82	42	40	88	73	27	●	●	●
Conf. I	83	42	72	88	79	21	●	●	●

As the last step, the privacy SPD level, SPD_P, of the sub-system has been calculated, and in order to simplify the most suitable configuration selection, a colour has been assigned based on the difference between SPD_P and the subsystem goal SPD_{Goal}.

Table 8 shows that e.g. conf. A,C satisfies Sc. 1, and conf. D-F satisfies Sc. 2. After examining the values, and before the most suitable configuration for each scenario can be selected, security *s* and dependability *d* values have to be evaluated using the same methodology.

This chapter explained the Multi-Metrics and shows its applicability using privacy as an example. In order to end up with a specific configuration which best satisfies the SPD_{Goal} of each scenario, is necessary to repeat the same process both with security and dependability. The final result, being the SPD_{System}, is a triplet with the exact security, privacy and dependability values obtained from the application of a given system configuration.

7 Evaluation

This section evaluates the applicability of Multi-Metrics approach and the presented methodology.

The outcome of the system analysis, presented in the earlier Sect. 6, showed various configuration options satisfying the privacy goals, SPD_P, defined by the application. As an example, Scenario 1 has a privacy goal of 80, which is satisfied by both configuration A ($SPD_P = 83$) and configuration C ($SPD_P = 77$). Following the same methodology, we can evaluate the system according to security and dependability. The result of the system evaluation will then be a triplet, e.g. $SPD_{Conf.A} = (42, 83, 64)$ for configuration A and $SPD_{Conf.C} = (22, 78, 53)$ for configuration C. The optimum configuration for a given scenario is then selected according to security and dependability analysis results.

Following the same example, we have defined an SPD_{Goal} of (50, 80, 60) for scenario 1. The most suitable configuration would be the first one, being $SPD_{Conf.A} = (42, 83, 64)$. Thus, the system, running within scenario 1 and configuration A, would have a SPD_{System} being (42, 83, 64) or (●, ●, ●).

The presented methodology considers all SPD aspects during the analysis of the most suitable configuration for each scenario. The picture obtained in the end shows under which SPD conditions the system will run for a given scenario and configuration. During the design phase of the Embedded System, the possible scenarios and configurations can be foreseen and the same analysis can be performed, providing security, privacy and dependability by design. The result will clarify if it is necessary to modify the design, of some system aspects, to satisfy the established goals.

Furthermore, for an existing system, the same analysis will provide a clear picture about the SPD_{System} level in operation. This analysis will identify which configuration options or system parts are not behaving as expected, thus, increasing the whole system risk. The early correction of misbehaving configuration options could prevent further consequences.

8 Conclusions

Embedded Systems evolved from isolated to highly interconnected devices, being the key elements of the Internet of Things. Their security, privacy and dependability (SPD) has been set aside, not fully considered during the design phase, but included as an add-on.

Moreover, each SPD aspect have been treated alone, without looking for a balanced solution.

In order to address this challenge, the Multi-Metrics methodology presented within this paper considers all SPD aspects together, both during the design and running phases. Its main advantages are the simplicity, Multi-Metrics is the core process used along all the steps, and scalability, it starts with component evaluation to jump over sub-systems and ends up with the entire system evaluation. The result is an overall SPD_{System} level, which makes it easy to understand under which configuration the system will perform as envisaged by the SPD_{Goal}.

The Multi-Metrics approach is applied for a smart vehicle use case, evaluating three scenarios with different privacy concerns. A total of nine system configurations are analysed, providing two configurations satisfying the privacy requirements of the first scenario, three satisfying scenario 2 and no configuration satisfying completely the SPD_{Goal} of scenario 3. Including security and dependability in the analysis provides a configuration suitable in answering the goal of an application.

The use case demonstrates the simplicity of the approach, as a single Multi-Metrics process is used during the whole system evaluation. Scalability is demonstrated, as simple and complex systems are evaluated equally, and each component, each sub-system and the resulting system can be classified through an (s, p, d)-triplet. In case of the metrics, further work needs to be done regarding their definition, components need to be characterized and their criticality established.

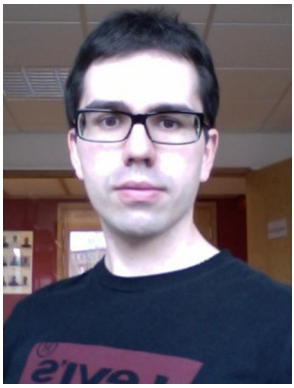
Acknowledgments The authors would like to thank their colleagues from the ARTEMIS project nSHIELD for the basics of the methodology, and the ongoing discussions on applicability. The work is financed in part by the JU ECSEL and the Research Council of Norway.

References

1. Alam, S., Chowdhury, M. M. R., & Noll, J. (2011). Interoperability of security-enabled internet of things. *Wireless Personal Communications*, *61*(3), 567–586.
2. new SHIELD (2014) nSHIELD. *New embedded systems architecture for multi-layer dependable solutions*. Retrieved September 9, 2014 http://www.newshield.eu
3. Manadhata, P. K., & Wing, J. M. (2011a). An attack surface metric. *Software Engineering, IEEE Transactions on*, *37*(3), 371–386.
4. Voas, J., & Miller, K. W. (1995). Predicting software's minimum-time-to-hazard andmean-time-to-hazard for rare input events. In *Software Reliability Engineering, 1995. Proceedings, Sixth International Symposium on, IEEE* (pp. 229–238).
5. Voas, J., Ghosh, A., McGraw, G., Charron, F., & Miller, K. W. (1996). Defning an adaptive software security metric from a dynamic software failure tolerance measure. In *Computer Assurance, 1996. COMPASS'96, Systems Integrity. Software Safety. Process Security. Proceedings of the Eleventh Annual Conference on, IEEE* (pp. 250–263).
6. Engler, D., Chelf, B., Chou, A., & Hallem, S. (2000). Checking system rules using system-specific, programmer-written compiler extensions. In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation, USENIX Association* (Vol. 4, pp. 1–16).
7. Engler, D., Chen, D. Y., Hallem, S., Chou, A., & Chelf, B. (2001). Bugs as deviant behavior: A general approach to inferring errors in systems code. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles, SOSP '01*, ACM, New York, NY, USA (pp. 57–72). doi:10.1145/502034.502041
8. Wagner, D., Foster, J. S., Brewer, E. A., & Aiken, A. (2000). *A first step towards automated detection of buffer overrun vulnerabilities*. New York, NY: NDSS.
9. Zhang, X., Edwards, A., & Jaeger, T. (2002). Using cqual for static analysis of authorization hook placement. In *USENIX Security Symposium*, (pp. 33–48).

10. United States Computer Emergency Readiness Team (US-CERT). (2014). *National cyber awareness system*. Retrieved September 9, 2014 https://www.us-cert.gov/ncas
11. National Institute of Standards and Technology (NIST). (2014). *National vulnerability database*. Retrieved September 28, 2014 http://nvd.nist.gov
12. MITRE. (2014). *Common vulnerabilities and exposures*. Retrieved September 28, 2014 http://www.cve.mitre.org
13. SecurityFocus. (2014). *Securityfocus*. Retrieved September 28, 2014 http://www.securityfocus.com
14. Browne, H. K., Arbaugh, W. A., McHugh, J., & Fithen, W. L. (2001). A trend analysis of exploitations. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on, IEEE*, (pp. 214–229).
15. Alves-Foss, J., & Barbosa, S. (1995). Assessing computer security vulnerability. *SIGOPS Operating Systems Review*, *29*(3), 3–13. doi:10.1145/206826.206829.
16. Beattie, S., Arnold, S., Cowan, C., Wagle, P., Wright, C., & Shostack, A. (2002). Timing the application of security patches for optimal uptime. *LISA*, *2*, 233–242.
17. Brocklehurst, S., Littlewood, B., Olovsson, T., & Jonsson, E. (1994). On measurement of operational security. *Aerospace and Electronic Systems Magazine, IEEE*, *9*(10), 7–16.
18. Howard, M. (2003). *Fending off future attacks by reducing attack surface*. Retrieved September 9, 2014 http://msdn.microsoft.com/en-us/library/ms972812.aspx
19. Bartel, A., Klein, J., Le Traon, Y., & Monperrus, M. (2012). Automatically securing permission-based software by reducing the attack surface: An application to android. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, ACM* (pp. 274–277).
20. Howard, M., Pincus, J., & Wing, J. M. (2005). Measuring relative attack surfaces. In D. T. Lee, S. P. Shieh, & J. D. Tygar (Eds.), *Computer Security in the 21st Century* (pp. 109–137). US: Springer.
21. Kurmus, A., Sorniotti, A., & Kapitza, R. (2011). Attack surface reduction for commodity os kernels: Trimmed garden plants may attract less bugs. In *Proceedings of the Fourth European Workshop on System Security, ACM*, p. 6.
22. Manadhata, P., & Wing, J. M. (2004). *Measuring a system's attack surface*. Tech. rep., DTIC Document.
23. Manadhata, P.K., & Wing, J.M. (2011). A formal model for a systems attack surface. In *Moving target defense, chap creating asymmetric uncertainty for cyber threats*, Vol. 54, (pp. 1–28). New York: Springer.
24. Stuckman, J., & Purtilo, J. (2012). Comparing and applying attack surface metrics. In *Proceedings of the 4th international workshop on Security measurements and metrics, ACM* (pp. 3–6).
25. Szefer, J., Keller, E., Lee, R. B., & Rexford, J. (2011). Eliminating the hypervisor attack surface for a more secure cloud. In *Proceedings of the 18th ACM conference on Computer and communications security, ACM* (pp. 401–412).
26. Public Interest, Inc. (2014). *Debian*. Retrieved October 15, 2014 http://www.debian.org
27. Red Hat, Inc. (2014). *Redhat*. Retrieved October 15, 2014 http://www.redhat.com
28. Manadhata, P. K., & Wing, J. M. (2005). *An attack surface metric*. Tech. rep., DTIC Document.
29. Manadhata, P. K., Tan, K. M., Maxion, R. A., & Wing, J. M. (2007). *An approach to measuring a system's attack surface*. Tech. rep., DTIC Document.
30. Manadhata, P., Wing, J., Flynn, M., & McQueen, M. (2006). Measuring the attack surfaces of two ftp daemons. In *Proceedings of the 2nd ACM workshop on Quality of protection, ACM* (pp. 3–10).
31. Kurmus, A., Tartler, R., Dorneanu, D., Heinloth, B., Rothberg, V., Ruprecht, A., et al. (2013). *Attack surface metrics and automated compile-time os kernel tailoring*. In NDSS.
32. Tartler, R., Kurmus, A., Ruprecht, A., Heinloth, B., Rothberg, V., Dorneanu, D., et al. (2012). Automatic os kernel tcb reduction by leveraging compile-time configurability. In *Proceedings of the Eighth Workshop on Hot Topics in System Dependability, ser. HotDep*, Vol. 12.
33. Krumm, J. (2009). A survey of computational location privacy. *Personal and Ubiquitous Computing*, *13*(6), 391–399.
34. Shokri, R., Theodorakopoulos, G., Le Boudec, J. Y., & Hubaux, J. P. (2011). Quantifying location privacy. In *Security and Privacy (SP), 2011 IEEE Symposium on, IEEE* (pp. 247–262).
35. Ma, Z., Kargl, F., & Weber, M. (2009). A location privacy metric for v2x communication systems. In *Sarnoff Symposium, 2009. SARNOFF'09, IEEE* (pp. 1–6).
36. Jatain, A., & Mehta, Y. (2014). Metrics and models for software reliability: A systematic review. In *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on, IEEE* (pp. 210–214).
37. Henkel, J., Bauer, L., Zhang, H., Rehman, S., & Shafique, M. (2014). Multi-layer dependability: From microarchitecture to application level. In *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*, ACM (pp. 1–6).

38. Weiner, M., Jorgovanovic, M., Sahai, A., & Nikolie, B. (2014). Design of a low-latency, high-reliability wireless communication system for control applications. In *Communications (ICC), 2014 IEEE International Conference on, IEEE* (pp. 3829–3835).



Iñaki Garitano is currently working as a postdoctoral fellow at UNIK-University Graduate Centre, Norway. He received the Ph.D. degree from the Department of Electronics and Computer Science, University of Mondragon in 2014 in the area of industrial control systems security. Prior to that he received the M.Sc. degree in Telecommunication Engineering from University of Mondragon. His current research interests include measurable security, privacy and dependability (SPD), Intrusion Detection Systems (IDS) and Internet of Things (IoT). He participated, and currently is involved, in research projects funded by the Norwegian Research Council, the Basque Government, the Spanish Government and the European Union.

Seraj Fayyad Ph.D. researcher at Movation AS and the University of Oslo/UNIK, he received his M.Sc. degree in computer engineering in the area of «reliable systems» from the University Duisburg-Essen, Germany. His research interests include IT security with concentration on measurable security for sensors in the Internet of People, Things and Services (IoPTS). He is involved in several international projects, including nSHIELD for measurable security in IoT systems, Citi-Sense-MOB for mobile air quality measurements.

Josef Noll is professor at the University of Oslo in the area of Wireless Network and Security. His work concentrates on personalised and context-aware service provisioning, and measurable security for the Internet of Things (IoT). He is also Head of Research in Movation, Norway's open innovation company. He is founding member of the Center for Wireless Innovation, the collaboration of 7 Universities/University colleges in Norway. He is involved in several international projects, including nSHIELD for measurable security in IoT systems, Citi-Sense-MOB for mobile air quality measurements, GravidPluss for mobile diabetes advise, and Ka-band propagation for polar regions. In the area of Internet of Things he was project leader of the Artemis pSHIELD project. Previously he was Senior Advisor at Telenor R&I in the Products and Markets group, and project leader of Eurescom's 'Broadband services in the Intelligent Home' and use-case leader in the EU FP6 'Adaptive Services rid (ASG)' projects, and has initiated a.o. the EU's 6th FP ePerSpace and several Eurescom projects. In 2008 he recieved the IARIA fellow award. He is editorial board member of four International Journals, as well as reviewer and evaluator for several national and European projects and programs.