Ensuring cyber-security in smart railway surveillance with SHIELD

Francesco Delli Priscoli and
Alessandro Di Giorgio*

University of Rome La Sapienza,
Via Ariosto 25, Rome, Italy
Email: dellipriscoli@dis.uniroma1.it
Email: digiorgio@dis.uniroma1.it
*Corresponding author

Mariana Esposito

Ansaldo STS,
Via Argine 425, Naples, Italy
Email: mariana.esposito@ansaldo-sts.com

Andrea Fiaschetti

University of Rome La Sapienza,
Via Ariosto 25, Rome, Italy
Email: fiaschetti@dis.uniroma1.it

Francesco Flammini

Ansaldo STS,
Via Argine 425, Naples, Italy
Email: francesco.flammini@ansaldo-sts.com

Silvano Mignanti

University of Rome La Sapienza,
Via Ariosto 25, Rome, Italy
Email: mignanti@dis.uniroma1.it

Concetta Pragliola

Ansaldo STS,
Via Argine 425, Naples, Italy
Email: concetta.pragliola@ansaldo-sts.com

Abstract: Modern railways feature increasingly complex embedded computing systems for surveillance that are moving towards fully wireless smart-sensors. Those systems are aimed at monitoring system status from a physical-security viewpoint, in order to detect intrusions and other environmental anomalies. However, the same systems used for physical-security surveillance are vulnerable to cyber-security threats, since they feature distributed hardware and software architectures often interconnected by 'open networks', like wireless channels and the internet. In this paper, we show how the integrated approach to security, privacy and dependability (SPD) in embedded systems provided by the SHIELD framework (developed within the EU funded pSHIELD and nSHIELD research projects) can be applied to railway surveillance systems in order to measure and improve their SPD level. SHIELD implements a layered architecture (node, network, middleware and overlay) and orchestrates SPD mechanisms based on ontology models, appropriate metrics and composability. The results of prototypical application to a real-world demonstrator show the effectiveness of SHIELD and justify its practical applicability in industrial settings.

Keywords: security; privacy; dependability; railway; surveillance; SHIELD.

Reference to this paper should be made as follows: Delli Priscoli, F., Di Giorgio, A., Esposito, M., Fiaschetti, A., Flammini, F., Mignanti, S. and Pragliola, C. (xxxx) 'Ensuring cyber-security in smart railway surveillance with SHIELD', *Int. J. Critical Computer-Based Systems*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Francesco Delli Priscoli was with Telespazio, Rome, Italy from 1986 to 1991. Since 1991, he has been with the University of Rome 'La Sapienza', Rome, Italy, where he is a Full Professor of Automatic Control. He was scientifically responsible for 32 projects mainly financed by the European Union, as well as for many national projects and cooperations with major industries. He is the author of about 200 papers, appearing in major international reviews, books, and conference proceedings. His main research is on networked systems. He is an expert in the field of resource management control for networked systems.

Alessandro Di Giorgio received his degree (Cum Laude) in Physics in 2005, and PhD in Systems Engineering from the University of Rome 'Sapienza' in 2010. He is currently a postdoctoral researcher in automatic control, working on original applications of control systems theory to smart grids and critical infrastructure protection, mainly in the context of national and European research projects. He is an author of about 40 papers and book chapters on these topics.

Mariana Esposito is a PhD in Computer and System Engineering. She is an expert in the field of security. Her research activities are focused on safety, reliability, and security of railway systems. Her activities are documented by publications in national and international conference proceedings.

Andrea Fiaschetti received his PhD in Systems Engineering and works as an Engineer at Thales Alenia Space Italia. Since 2013, he has been an honorary fellow at the University of Rome 'La Sapienza', where his research interests are in the field of applied automatic control, pursuing a cross-fertilization between control theory and computer science. His major achievement is the formalisation of the so-called 'composable security theory', aimed at establishing the foundations of next generation embedded systems. He is an author of several conference papers, journal papers and books contributions on these topics.

Silvano Mignanti holds a PhD in Systems and Control Engineering. He has more than 11 years of experience in computer science, ITC, service management and security for networked systems. He is an author of several publications on these topics.

Concetta Pragliola received her Laurea and Doctorate degrees in Electronic Engineering from the University Federico II of Naples in October 1985. From January 1987 to October 2001, she has worked at Ansaldo Transporti. From November 2001 to November 2006, she has worked in Elsag as an Account Manager. She is currently with the innovation unit of Ansaldo STS working on the design of security systems.

1 Introduction

Embedded systems (ES) employed in cyber-physical monitoring and control applications feature increasingly complex (i.e., large, distributed, heterogeneous) architectures and strict requirements about security, privacy and dependability (SPD). In order to manage such complexity, it is essential to develop new frameworks allowing the management of SPD requirements in a way that is both effective and efficient. Several research efforts have been performed to address the resilience of ES through hardware/software fault/attack-tolerance and dynamic reconfiguration; however, none of those approaches address the overall issue in a way that is integrated, cohesive and holistic, using semantic modelling, ontologies and control systems theories implemented through appropriate middleware and SPD-technologies, that is the scope of the SHIELD framework presented in this paper (see references Esposito et al., 2013; Delli Priscoli et al., 2012a; pSHIELD Project, http://pshield.unik.no/wiki/PSHIELD-public; nSHIELD Project, http://www.newshield.eu/.).

Among the critical applications of ES, there are the ones addressing physical security that is the protection against intentional attacks of malicious nature like thefts, sabotage, terrorism, etc. The issue is very relevant in the context of critical infrastructure security (e.g., Casola et al., 2012b; Canale et al., 2012), where large, distributed and heterogeneous surveillance systems are employed (Flammini, 2011; Di Giorgio and Liberati, 2011). Those systems are used to monitor the environment to detect physical threats and activate appropriate response countermeasures. As a matter of fact, the same ES used for physical security monitoring can be subject to cyber-security attacks aimed at deactivating or spoofing intrusion detection and access control devices, or at getting private information about user data or video footage.

One of the nowadays most relevant domains of critical infrastructure protection is railway and mass-transit surveillance, since for their nature (open systems moving a very large number of passengers) rail-based transit systems are attractive targets for adversaries ranging from thieves to terrorists (Hartong et al., 2008). In fact, there has been a growing interest of railway operators in physical security information management systems (Bocchetti et al., 2009) (see Figure 1), as well as in novel smart-sensing platforms (Flammini et al., 2010; Hodge et al., 2015). Unfortunately, most of the sensors nowadays available for railway surveillance feature weak information security and resilience mechanisms (if any), that are difficult to measure, integrate and control,

especially considering the requirements of easy scalability, expansion and maintainability requested by the end users.

Figure 1 A control room for the physical security information management (see online version for colours)

The scope of this paper is to present the general features of the SHIELD framework and to show an example case-study application to the cyber-security of a network of smart wireless devices used to monitor a critical railway asset.

The rest of this paper is structured as follows. Section 2 and its sub-sections provide a general description of the SHIELD framework, focusing on metrics, semantic models, middleware architecture and composability mechanisms. Section 3 and its sub-sections describe the railway security demonstrator, its reference architecture, the involved SHIELD prototypes, the case-study scenario, and demonstration results. Finally, Section 4 provides conclusions and hints about future developments.

2 The SHIELD framework

In recent years, ES technologies have seen an exponential diffusion in our daily life, from business environment to personal entertainment, mainly due to the high availability of low-cost computational capabilities.

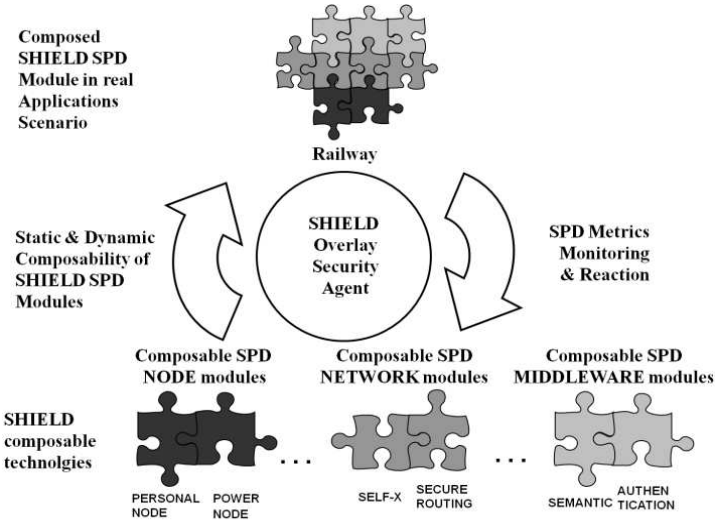
The pervasive presence of ES has therefore raised new challenges and problems that need to be properly addressed through strategic initiatives. In this perspective, the European Commission, within the seventh framework program (FP7) has established the ARTEMIS JU (today known as ECSEL), a joint undertaking in charge of defining and implementing a roadmap that will drive the growth of ESs industry towards really effective objectives (ECSEL JU, http://www.ecsel-ju.eu/). One of these objectives is the development of new technologies and/or strategies to address SPD in the context of ESs, with major impacts on all those applications involving safety, reliability and security.

To properly address this challenge, a restricted pool of academic and industrial researchers has created the SHIELD roadmap, whose output was the SHIELD Framework, an innovative methodology to address SPD in complex system as a ‘built in’ feature, rather than ‘add-on’ functionality.

In a nutshell, a complex system is seen as a mixture of atomic elements performing specific tasks (that can be SPD relevant): the main purpose of the SHIELD methodology is to enable composability of these atomic functionalities. A trivial representation is provided in Figure 2.

The SHIELD SPD modules can be represented as pieces of a puzzle, which perfectly fits each other thanks to common interfaces. Each module implements a SPD technology or a specific SPD functionality. As an example, in Figure 2 at *node level* there are two modules: personal node and power node technologies, at *network level* there are two functionalities: self-x algorithms and secure routing, and at *middleware level* there are two services: semantic management and authentication.

Figure 2 SHIELD composability

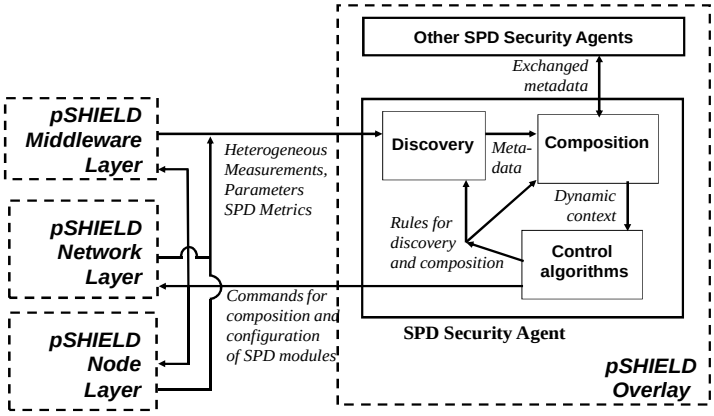


These modules, belonging to different SPD layers (node, network or middleware), can be composed statically or dynamically by the *SHIELD overlay*. Furthermore, individual SHIELD SPD modules can be replaced once the measured SPD metrics do not satisfy the required SPD levels. Indeed the SPD metrics are continuously monitored by the security agents and in case of failure, the security agent reacts by discovering, composing and configuring the available SPD modules.

The SHIELD reference architecture is depicted in Figure 3 in a more formal representation (as already described in Fiaschetti et al., 2014, 2012), with the indication of the technological enablers: *metrics*, *ontologies* (or metadata) and *overlay*.

The complex system is divided into its atomic elements at node (i.e., hardware), network (i.e., communication) and middleware (i.e., software) named 'SPD functionalities' or 'SPD technologies'. Then, on top of that, SHIELD introduces an overlay of security agents which will be in charge to implement the key *composability* concept (see Figure 3). The security agents will be placed in appropriate network entities to be properly selected according to specific criteria which take into account the considered scenario.

Figure 3 SHIELD functional architecture



Each security agent monitors a set of properly selected measurements and parameters taken at any of the three above-mentioned layers (see the arrows labelled as *measurements* in Figure 3). These heterogeneous measurements and parameters are converted by the security agents in *homogeneous metadata* by extensively using properly selected semantic technologies; the use of homogeneous metadata makes easy the metadata exchange among different security agent (Figure 3). Each security agent, thanks to metadata homogeneity, can aggregate the available metadata (the ones relevant to monitored measurements and parameters, as well as the ones coming from other security agents), in order to deduce aggregated metadata which form the so-called *dynamic context.* The latter is used as basic input for a set of *control algorithms* responsible of dynamically deciding which SPD modules have to be composed and enabled/disabled at any of the three above-mentioned layers, as well as how the activated modules have to be configured in order to achieve the desired SPD level. These decisions are enforced in the interested SPD modules lying at the three above-mentioned layers (see the arrows labelled as *commands* in Figure 3). The above-mentioned control algorithms are also in charge of possibly updating the rules to form the dynamic context (i.e., which measurements and parameters have to be monitored, which metadata have to be exchanged with other security agents, how the available metadata have to be aggregated, etc.).

Note that the strength of the presented composability concept lies in the possibility of *jointly* deciding at the inter-layer manager, basing on information gained at all layers, which SPD provisions have to be performed at each layer in order to achieve the *overall* desired SPD level. This approach has the evident advantage of allowing taking SPD provisions which are coordinated among the different layers and of permitting to decide on these provisions on the basis of aggregated information coming from all layers.

In the following sections, the main enabling technologies for this architecture will be described: metrics, semantic models and middleware.

2.1 The SHIELD metrics

SPD metrics is the SHIELD key issue. The SHIELD project has identified static and dynamic SPD metrics driven by the requirements coming from applications, at each of the considered layers, as well as for the overall system. Then, SHIELD identifies the ES desired SPD level at each layer and for the overall system with respect to these metrics.

The 'SHIELD attack surface metrics' is an approach developed in order to compute the SPD level in the SHIELD framework. The approach is an integration of three different methods: 'attack surface metric' (Manadhata and Wing, 2010), 'the open source testing methodology manual (OSSTMM) 3' (Herzog, 2010) and 'common criteria evaluation methodology (CEM)' (Common Criteria, 2012). Such integration allows expressing the SPD level as a plain number.

An *attack surface* is a set of modes by which an attacker can entry in contact with a system and cause a disaster or a failure.

The SHIELD metrics integrate dependability and security concepts. It considers the threat as the origin of the chain 'fault → error → failure' as well as the potential for abusing protected assets.

The malicious human activity or non-malicious events are addressed at the entry and exit points of the system. The entry and exit points are characterised by three factors: porosity, controls, and limitations (Herzog, 2010). The characteristics of entry and exit points define the likelihood of being exploited by attackers. The measurement is the total contribution of porosity, controls and limitations.

A threat has the capacity to subvert the security or the dependability of a system and in order to be effective; it shall interact directly or indirectly with the asset. So the aim is to separate the threat from the asset in order to avoid the interaction (e.g., total separation means SPD level = 100). Any protection increases the SPD level and can be activated by controls to the asset, in order to reduce the impact a threat.

During the analysis phase (a sort of 'vulnerability assessment'), it is important is to identify the possible interactions. This parameter is called 'porosity'. The porosity reduces the separation between a threat and an access. It is characterised by three elements: complexity, access and trust.

Each point of interaction (access) reduces the security and then the SPD level. The increase of porosity is the decrease in SPD and each pore is a complexity, access or trust. In detail:

- complexity: number of SPD critical components
- access: number of possible interactions with the system
- trust: access not threatening system security.

For each access pore identified, damage potential-effort ratio need to be computed to have a consistent measure of the introduced lack of separation. Access pores do not equally contribute to system porosity since they are not equally likely to be exploited by attackers.

Controls reduce the interaction between threat and assets. There are two main categories of controls, ‘interactive’ and ‘process’, for a total of 12 types of controls.

Interactive Controls are directly related to complexity, access, or trust interactions, and they influence them. The categories are the following:

- Authentication is a control through the challenge of credentials based on identification and authorisation.
- Indemnification is a control through a contract between the asset owner and the interacting party. This contract may be in the form of a visible warning as a precursor to legal action if posted rules are not followed, specific, public legislative protection, or with a third-party assurance provider in case of damages like an insurance company.
- Resilience is a control over all interactions to maintain the protection of assets in the event of corruption or failure.
- Subjugation is a control assuring that interactions occur only according to defined processes. The asset owner defines how the interaction occurs which removes the freedom of choice but also the liability of loss from the interacting party.
- Continuity is a control over all interactions to maintain interactivity with assets in the event of corruption or failure.

Process controls define defensive processes. These controls do not directly influence interactions; rather, they protect the assets once the threat is present. The categories are the following:

- non-repudiation is a control which prevents the interacting party from denying its role in any interactivity
- confidentiality is a control for assuring an asset displayed or exchanged between interacting parties cannot be known outside of those parties
- privacy is a control for assuring the means of how an asset is accessed, displayed, or exchanged between parties cannot be known outside of those parties
- integrity is a control to assure that interacting parties know when assets and processes have changed
- alarm is a control to notify that an interaction is occurring or has occurred.

The classes of limitation are listed in the following:

- vulnerability: denying access to authorised entities (people or processes), allow privileged access to unauthorised entities, or allowing unauthorised entities to hide assets or themselves
- weakness: abusing or nullifying the effects of the interactivity controls
- concern: disrupting or reducing the effects of the process controls

- exposure: unjustifiable action or error that provides direct or indirect complexity of targets or assets
- anomaly: unexpected error or flaw.

The process of obtaining the SPD level for the whole system is composed by several steps. The starting point is the analysis of the system and the recognition of the components. In order to simplify the computation and to reduce the number of system states, some components are merged according to SPD functionalities generated from their cooperation, whether they:

- have a common physical boundary
- could work together to identify a logical functionality.

In this way, it is possible to apply the Attack surface metric composition rules to obtain all the possible SPD levels for the system.

2.2 The SHIELD semantic framework

As outlined in the architecture (Figure 3) and widely justified in the pilot phase of the project (see Fiaschetti et al., 2011; Suraci et al., 2012), the middleware modules (and above all the security agent) need a proper semantic framework to model and elaborate all the information exchanged among the system and relevant to implement the composability mechanism. This framework is composed by:

- an ontology, to model technology independent information (i.e., metric value)
- a domain database, to model technology and scenario dependent information.

The SHIELD ontology is a simple translation of the 'attack surface' and 'porosity' concepts. Since the surface is a function of the amount of interfaces to the external world (access), interactions between components (complexity) and internal/external interactions with no direct impact on security (trust), these concepts are included in the 'system' part of SHIELD ontology (Figure 4). These attributes are represented by a number, so the generic SPD functionality brings a numeric contribution for each of these attributes. These values hold both for the system and for each additional SPD functionality.

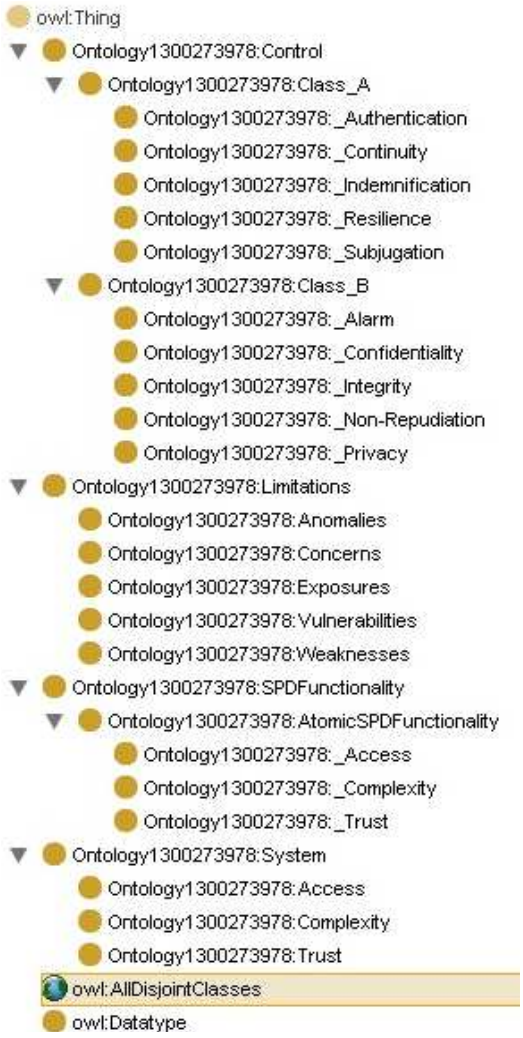
As described in the previous section, each vulnerability, identified by the number of 'accesses', can be counteracted by means of specific controls. Controls are classified in class A and class B, and can be translated into ontology as well (Figure 4).

Each SPD functionality brings into the system one or more controls. Each control, once activated, can be affected by a set of limitations that are included in the third section of the SHIELD ontology (Figure 4).

Each element depicted in this ontology can be:

- a simple number (i.e., two integrity controls, …)
- an element itself (i.e., CRC control, hash integrity control, …).

Figure 4 SHIELD ontology (see online version for colours)



The resulting XML file is reported in Figure 5 and, due to the very high-level information represented; its size is surprisingly small: about 2 kB that is suitable for low resource environments.

Each SHIELD component/device can implement a set of SPD functionalities, introducing controls, limitations, vulnerabilities and all those information are stored into an XML file.

When all the XML files are collected by the security agent, they are put together to build one single XML file representing the security level of the overall system. This composition has to cope with:

- interfaces
- contracts
- exceptions.

Figure 5 Sample SHIELD XML (see online version for colours)

```
<metrics>
<vulnerabilities>
<basic>0</basic>
<e_basic>0</e_basic>
<moderate>0</moderate>
<high>1</high>
<beyond_high>1</beyond_high>
</vulnerabilities>
<limitations>
<anomalies>2</anomalies>
<concerns>4</concerns>
<exposures>1</exposures>
<weaknesses>6</weaknesses>
</limitations>
<classA>
<authentication>14</authentication>
<indemnification>1</indemnification>
<resilience>4</resilience>
<subjugation>29</subjugation>
<continuity>4</continuity>
</classA>
<classB>
<non_Repudiation>11</non_Repudiation>
<confidentiality>2</confidentiality>
<privacy>1</privacy>
<integrity>3</integrity>
<alarm>6</alarm>
</classB>
<complexity>4</complexity>
<trust>6</trust>
<accessesList>
<dp>4</dp>
<ef>2</ef>
<num>1</num>
</accessesList>
<accessesList>
<dp>4</dp
<ef>4</ef>
<num>2</num>
</accessesList>
<SPD>84.95</SPD>
</metrics>
```

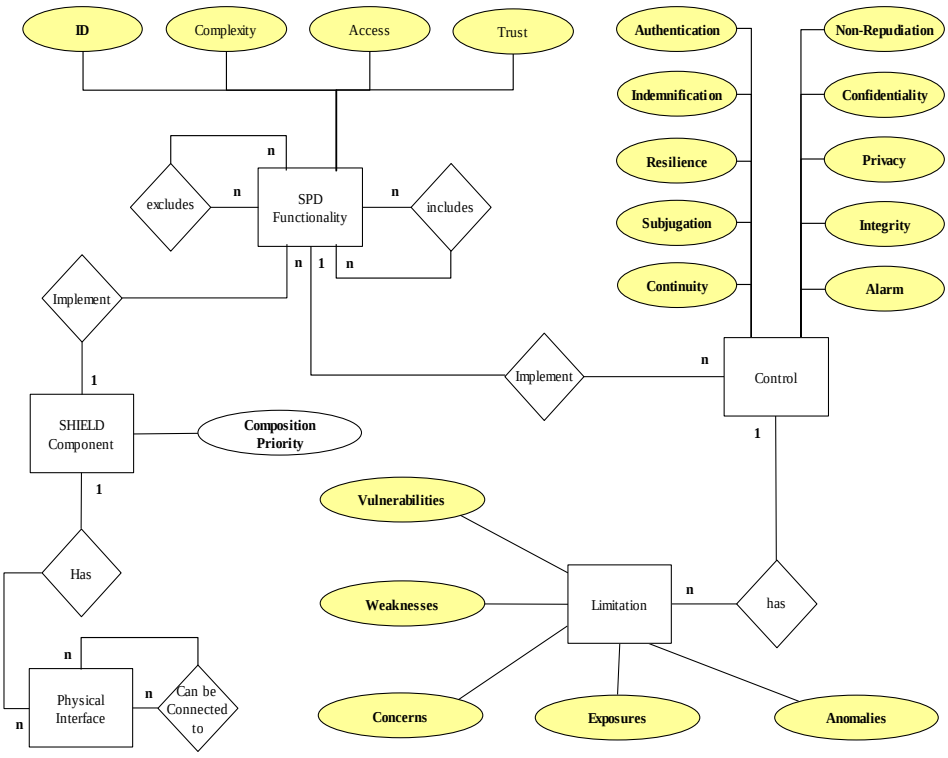
Since those information are strictly linked to the current environment and operating domain, it is reasonable to find the composition information and rules in the domain database. The role of the domain database (or library) is to tailor the technology-independent information to the specific application scenario.

This library contains all the refinements necessary to tailor the abstract components to the specific scenario requirements, as well as to perform metrics composition. In particular, it contains:

- a replica of the XML information (relevant for metrics computation)
- a list of numerical values for the metrics attribute of the defined ontology
- a list of functional dependencies between the SPD functionalities (mutual inclusion and mutual exclusion)
- a list of connection interfaces (i.e., topological information) to identify internal and external interfaces after elements coupling
- a 'composition priority' attribute indicating the order in which different elements should be composed.

The E-R representation of this DB is reported in Figure 6 (in yellow the parts that can override the information already included in the XML ontology).

Figure 6 SHIELD domain dependent library E-R diagram (see online version for colours)



This library can be also referred to as ‘context’ and it is used by the overlay during the composition process in this way:

Step 1 At first the ontologies (XML) are retrieved by means of discovery services.

Step 2 Ontologies (XML) are updated by means of context information.

Step 3 According to functional dependencies (inclusion/exclusion) only compatible SPD functionalities are considered for composition. The SPD functionalities may also be forced or deleted by proper, domain dependent, policies (i.e., a policy may force ciphering).

Step 4 Following a priority order given by the priority field, individual XMLs are couples iteratively, to derive a single XML file starting from two atomic files. The result is then coupled with another atomic XML, or with an XML resulting from a previous coupling. The process is repeated until only one XML is obtained, that reports the information about the whole system. The composition rules for the quantification of the resulting metric value are reported in the following page.

Step 5 The resulting file contains a certain amount of possible ‘variation’ of the metrics relevant parameters (e.g., *n* vulnerabilities, *m* weakness, and so on). By varying these values from 1 to *m* or from 1 to *n*, all the possible condition in which the system may operate are identified, with the associated metric value. The conditions are named ‘states’. The states may also be forced or deleted by proper, domain dependent, policies (i.e., a policy may remove all the states involving ciphering). The possible solutions for the composition problem are then identified.

Step 6 At runtime, according to the current condition of the system, the solution for the composability problem is computed.

The rules of metrics composition are used to determine the value of the SPD level for a system (scenario) in a given state, once calculated the SPD level that the various components (prototypes) that constitute it can take (for each state of the prototype).

At this purpose, first of all, starting on system (scenario) architecture, it is necessary to define by successive steps how the various components (prototypes) are connected physically creating the elements and/or sub-systems, until all join in the composition of the system (scenario) proposed.

The starting points for this calculation are the XML file that each prototype provides. Therefore, once defined the basic rules for the composition of two files associated with the two prototypes is possible to automate these rules for the calculation of SPD levels defined in the XML file that represent the various states of the system from those of the components (prototypes) that constitute it.

In Tables 1 to 3, composition rules for each field of the XML file are reported, with the rationale for each of them and notes to handle exceptions.

Table 1 Porosity composition rules

Element	*Composition rule*	*Rationale*	*Notes*
Complexity	Sum	Must consider all critical elements which failure might not be tolerated by system architecture	If the same element is critical for more than one component, it must be considered only once. If a single component has more than one critical element, it must be considered as 1 in the composition.
Access	Sum (for the different types)	Must consider all possible accesses to the composition of components	If one access is common to both components, it must be considered as 1. If one access of the first component belongs also to other components and it is internal to the composition of components (with a relationship of trust), then these accesses must not be considered and the Trust element must be incremented by one.
Trust	Sum	Must consider each relationship that exists where the system accepts interaction from its components or from another system	If one access of the first component belongs also to other components and it is internal to the composition of components (with a relationship of trust), then these accesses must not be considered and the Trust element must be incremented by one.

Table 2 Controls composition rules

Element	*Composition rule*	*Rationale*	*Notes*
Confidentiality Privacy Authentication Resilience Integrity Non-repudiation Subjugation Continuity Indemnification Alarm	Sum (for the different control categories)	Must consider all controls that counteract threats and their effects.	

Table 3 Limitations composition rules

Element	*Composition rule*	*Rationale*	*Notes*
Exposure	Sum	Must consider all unjustifiable actions, flaws, or errors providing direct or indirect visibility of targets or assets within the chosen scenario interface	If the same element represent an exposure for more than one component, it must be considered only once
Vulnerability	Sum (for the different rating)	Must consider all possible flaws or errors that: a deny access to assets for authorised people or processes b allow privileged access to assets to unauthorised entities c allows unauthorised entities to hide assets or themselves	
Weakness	Sum	Must consider all possible flaws or errors that disrupt, reduce, abuse, or nullify the effects of the five interactivity controls: authentication, indemnification, resilience, subjugation, and continuity	
Concern	Sum	Must consider all possible flaws or errors that disrupt, reduce, abuse, or nullify the effects of the flow or execution of the five process controls: non-repudiation, confidentiality, privacy, integrity, and alarm.	
Anomaly	Sum	Must consider all unidentifiable or unknown elements which cannot be accounted for in normal operations, generally when the source or destination of the element cannot be understood.	If more than one component considers the same anomaly, it must be counted only once

2.3 The SHIELD middleware

Figure 7 depicts the reference nSHIELD middleware and overlay architecture that is the software layer in charge of implementing the services necessary to perform the discovery (Casola et al., 2012a) and composition of SPD functionalities.

The security agent is the core of the SHIELD system, since it implements the control algorithms that drive the composability. Expandability of such framework is obtained by enabling communication between security agents controlling different sub-systems through a proper overlay interface. Therefore, the presence of more than one SPD security agents is justified by the need for solving scalability issues in the scope of system-of-systems (exponential growth of complexity can be overcome only by adopting a hierarchical policy of *divide et impera*).

Figure 7 SHIELD middleware and overlay

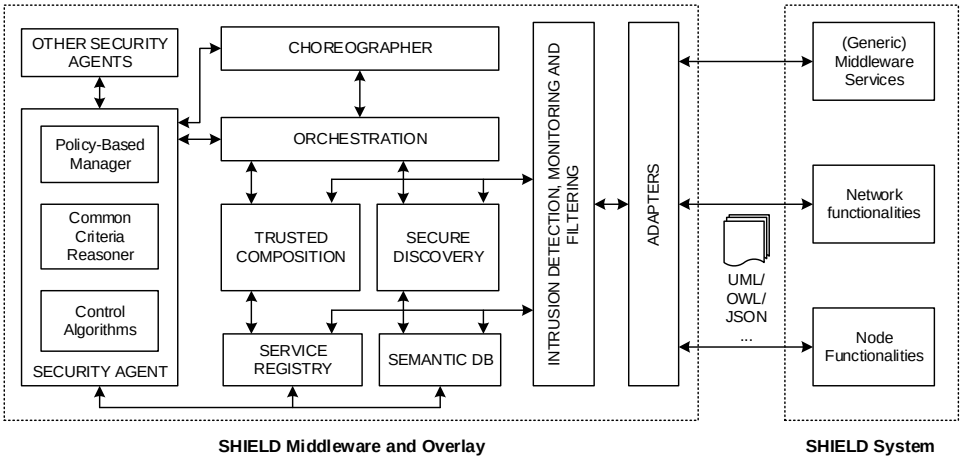
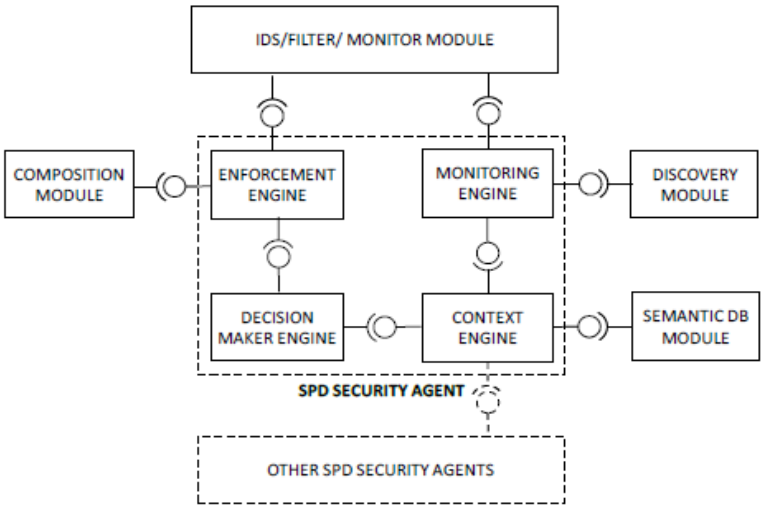


Figure 8 Security agent architecture



A zoom on the internal architecture of the security agent is provided in Figure 8:

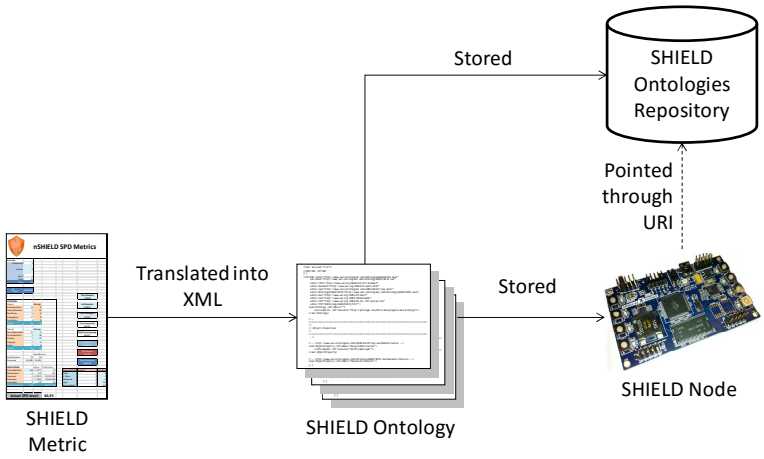
- The *monitoring engine* is in charge to interface the Overlay layer with the Middleware layer, to retrieve sensed metadata from heterogeneous SHIELD devices belonging to the same subsystem, to aggregate and filter the provided metadata and to provide the subsystem situation status to the context engine.
- The *context engine* is in charge to keep the situation updated as well as to store and keep updated any additional information exchanged with other SPD security agents that are meaningful to keep track of the situation context of the controlled SHIELD subsystem. The situation context contains both status information and configuration information (e.g., rules, policies, constraints, etc.) that are used by the decision maker engine.

- The *decision maker* engine uses the valuable, rich input provided by the context engine to apply a set of adaptive (closed-loop or rule-based) and technology-independent algorithms. The latter, by using (as input) the above-mentioned situation context and by adopting appropriate advanced methodologies able to profitably exploit such input, produce (as output) decisions aiming at guaranteeing, whenever it is possible, target SPD levels over the controlled SHIELD subsystem.
- The decisions mentioned above are translated by the *enforcement engine* into a set of proper enforcement rules actuated by the SHIELD middleware layer all over the SHIELD subsystem controlled by the considered SPD security agent.

2.4 How does composability work in five steps

Summing up all the concepts described so far, the SPD composability can be achieved through several simple steps. The first step is depicted in Figure 9.

Figure 9 Shield ontology rationale (see online version for colours)



In the SHIELD system, the security agent needs a mean to have both qualitative and quantitative information about system elements needing to be composed. The SHIELD ontology is stored into an XML file that contains the translation of the 'attack surface metric'.

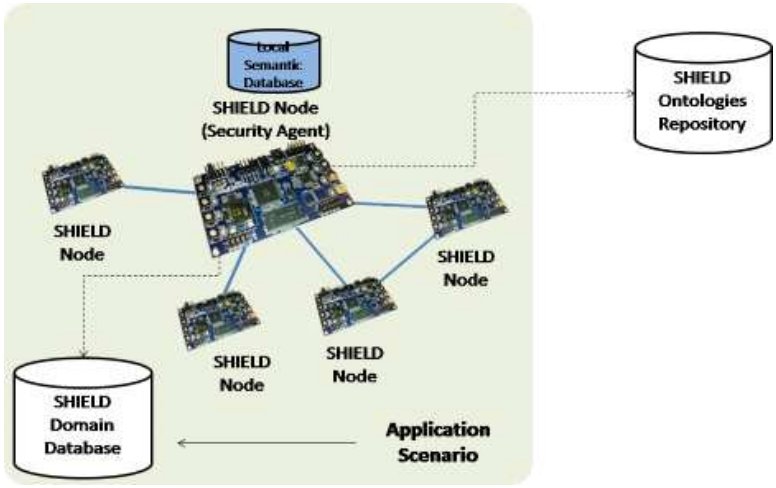
The assumption is that a system expert fills in a sheet containing all the information needed to compute the attack surface metric. These sheets are then translated into XML (in order to ease information storage and parsing) and stored into a local memory of the component or on a remote SHIELD ontology repository accessible through the internet using a specific uniform resource identifier (URI) [in this respect, recent advancements in the future internet field might prove useful (Delli Priscoli et al., 2012b; Bruni et al., 2016)].

In the second step (depicted in Figure 10), the SHIELD System is deployed into a specific application scenario, with the aim of implementing some security related end-to-end behaviour. As far as the elements are put in place, domain experts populate a

domain database with a set of information necessary to the security agent to tailor its decisions. In particular in this domain DB one can find:

- *Architectural/topological* information related to the system deployment, with specific focus on:
  1 component interfaces (necessary for metric composition)
  2 composition hierarchy/order (also necessary for metric composition)
  3 functional dependencies (used by the composition engine to activate ancillary services).
- *Policies and constraints* to drive the control action by:
  1 forcing specific system configurations under specific circumstances
  2 erasing specific system configurations from the available ones.

Figure 10 SHIELD framework deployment (see online version for colours)



In operating conditions, the SHIELD system is supposed to have access to several data sources: the XML stored into the ES (or alternatively on the ontologies repository) and the domain database. Those information are collected by the security agent and the derived data are stored into a local DB (i.e., a volatile memory of the software component).

When the SHIELD framework is initialised or whenever variations in environmental or architectural conditions are detected, the discovery engine starts monitoring the system to collect the XML from the SHIELD Nodes or retrieve information from the domain DB. In this way, the security agents' knows the list of all the available elements as well as (domain DB) their architectural dependencies and composition hierarchy. Using those information, it can iteratively couple the elements until a single metric value is obtained for the overall system. At this stage, Policies may also override or delete specific configurations that are not permitted.

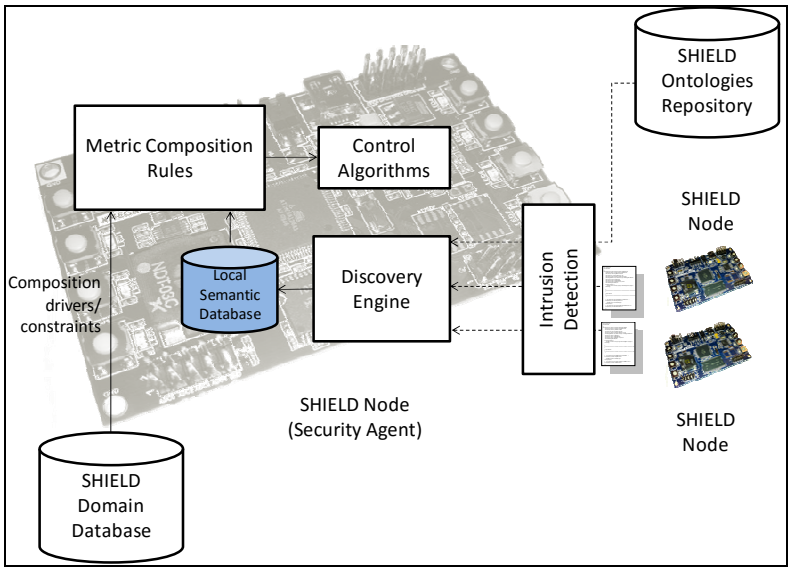
In case of multiple states for the same configuration, the corresponding metrics are computed. The result is then a vector of 1 to *n* elements, representing the n system

configurations and the corresponding metric value. This information is then propagated to the control algorithm module.

Please note that, in order to prevent malicious attacks or flooding during the discovery process, the middleware is protected by a robust intrusion detection system.

The computation of the SPD composition solution is a problem of choosing a configuration that implements the desired SPD level. This is trivial in case of single admissible solution, but it can be difficult in case of several possible states and configurations. In the SHIELD research project, several approaches have been investigated based on hybrid automata, coloured Petri nets, etc., with the most effective one being an optimisation algorithms that – given as inputs all the possible configurations – is able to compute the configuration that maximises or minimises an objective function, that is the distance between current and desired SPD levels.

Figure 11 SHIELD discovery and 'baseline' composition (see online version for colours)



A second way to manage SPD composition is by using policy-based management (PBM). PBM works in parallel with the security agent and drives the composition according to pre-defined, deterministic rules that force specific system behaviour and components activation. The PBM solution is more suitable when safety aspects are involved, or when domain security is driven by reference standards. Such a dual approach is depicted in Figure 12.

Once the selected configuration (i.e., solution) is computed, the composition engine enforces the decisions back to the system by means of:

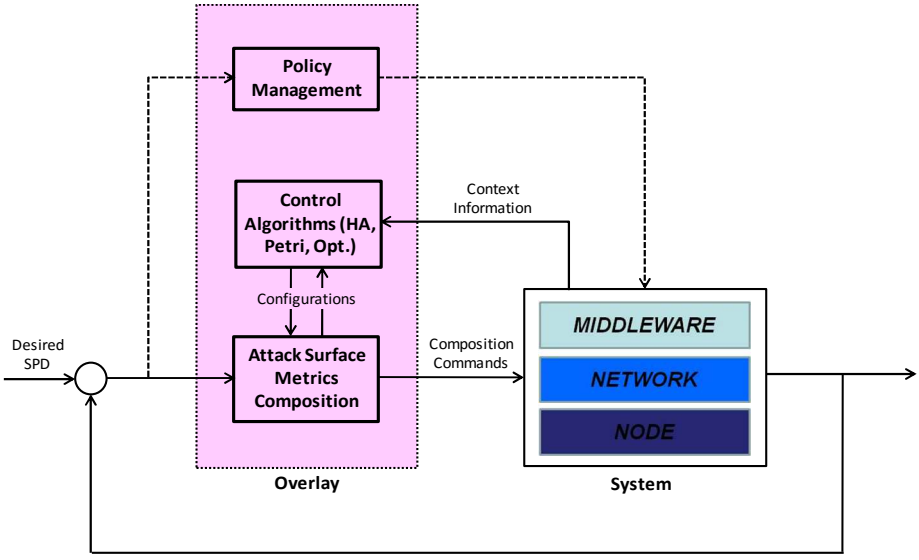
1 composition commands sent to SHIELD compliant nodes

2 proper adapters, in case of legacy ES.

The composition command is simply a list of commands to be executed in order to activate services or HW configurations. The mapping between the domain command and the security agent high-level decisions is reported in the domain database: for example, if

the SHIELD system has to interact with a Railway server to orchestrate services, then the list of server commands with associated 'services' is stored in the domain database, so that the discovery engine is able to interact with them.

Figure 12 Composition problem solution (see online version for colours)



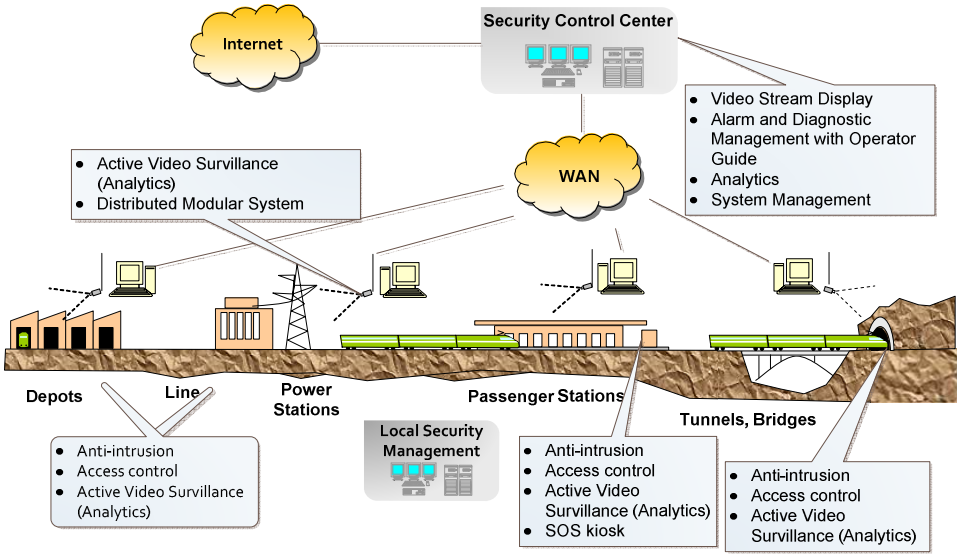
3 The railway security demonstrator

3.1 Reference architecture

A railway security system is aimed at detecting and possibly counteracting physical threats like abnormal behaviours, thefts, vandalism, sabotage, etc. A railway security system (Bocchetti et al., 2009; Flammini, 2011) is composed by several types of devices including access control and intrusion detection systems, cameras and other smart-sensors for environmental monitoring (Figure 13). In recent years, even wireless and low-power smart-sensors have started to be used for their flexibility as well as unique and convenient features: low hardware cost, low-power draining, reduced or no cabling, on-board programming to provide distributed 'intelligence', easy to obtain mesh-network topologies and multi-hop routing, possible 'plug and play' installation (Hodge et al., 2015).

However, the usage of wireless smart-sensors poses novel threats to information security due to possible cyber-attacks to stored and transmitted data, at any link of the path connecting sensors to the control centre. In particular, certain buildings, namely 'shelters', are especially sensible targets, since they can contain valuable and possible critical equipment used for signalling and telecommunications. The trackside location of shelters is typically far from stations and depots, hence remote and automated monitoring of possible cyber-physical attacks is essential.

Figure 13 Typical architecture of a railway security system (see online version for colours)



Shelters can be equipped with sensors measuring environmental parameters, such as temperature, humidity, vibration, light, etc., as well as motion detection cameras used to detect intrusions and visually verify the consistency of other alarms. Environmental monitoring is important for both non-intentional and intentional (human-made) threats like flooding, fire/overheating, unauthorised door opening, manumissions, etc. In case of wireless inter-sensor communication, wireless sensor networks (WSN) messages can be subject to the threats affecting 'open communication channels' (repetition, deletion, insertion, re-sequencing, corruption, delay, masquerade), as defined in the CENELEC EN50159 standard specification for railway applications.

The reference architecture of the demonstrator is shown in Figure 14. The architecture is composed by an operation control centre in which the SHIELD-middleware server and client are installed. Alarms and SPD-state variations are managed by the middleware server and monitored by the operators through the middleware client.

The demonstration scenario consists of a shelter monitoring system featuring the following sensor types:

- WSN motes
- Smart cameras.

Those devices are vulnerable to cyber-attacks at the network level, such as *black-hole attacks* (Ramaswamy et al., 2003) and *bad-mouthing attacks* (Vijaya and Selvam, 2013). Black hole attack in a network implies that one or more malicious nodes would partially or fully drop data packets being routed through it causing disruptions in the normal data flow in the network. Malicious node advertises itself as the best route towards the sink node just like other sensor nodes. The sender nodes select the malicious node as their parent node (next in line node in the routing topology) and start forwarding their data packets; these data packets are then dropped. In a bad mouthing attack an attacker gives negative feedback on a node in order to lower or destroy its reputation.

Figure 14 Railway security demonstrator reference architecture (see online version for colours)

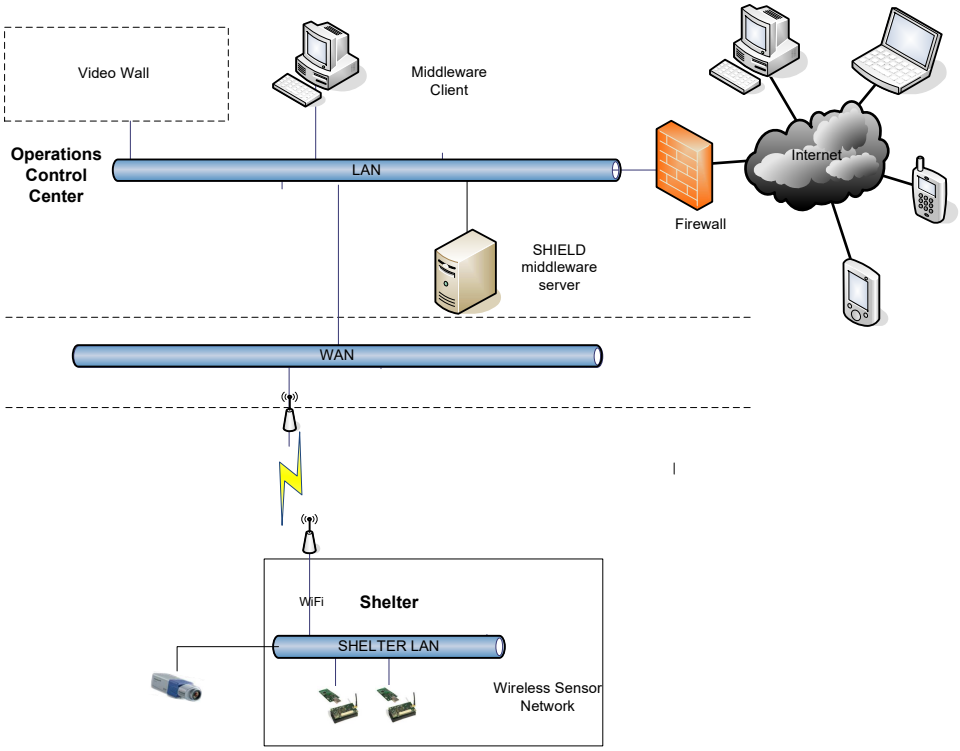
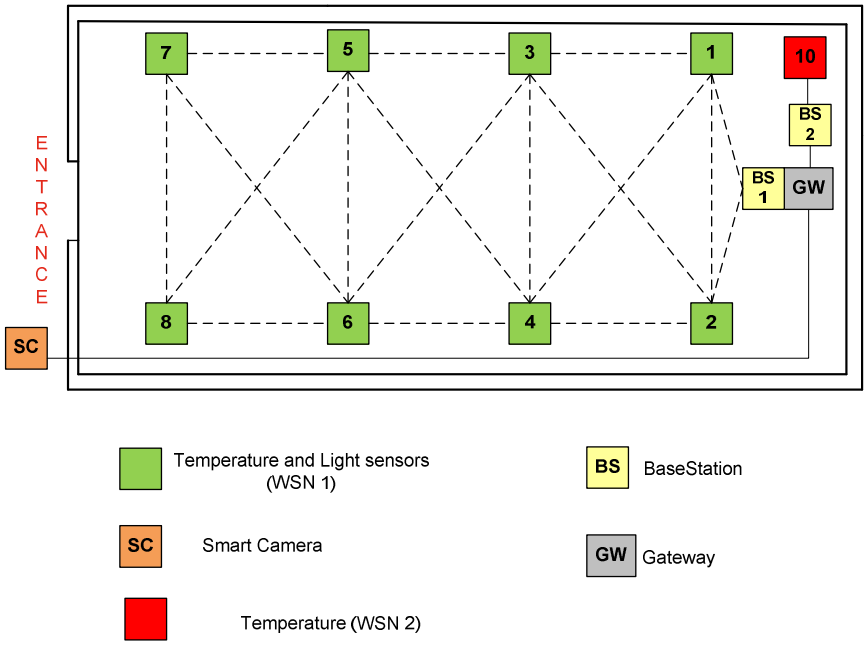


Figure 15 Network topology in the shelter (see online version for colours)



In the shelter, the following sensors are installed:

- at the entrance, smart-cameras are installed to detect physical intrusions
- inside the shelter, two diverse WSNs are installed: WSN_1 (in green) measures temperature and light, while the WSN_2 (in red) only measures temperature.

WSN_1 and WSN_2 feature different hardware and software in order to provide diverse redundancy. The topology of the network is shown in Figure 15. The gateway acts as the link through which the information is sent to the control centre.

3.2 Involved SHIELD prototypes

The scenario uses the following SHIELD prototypes:

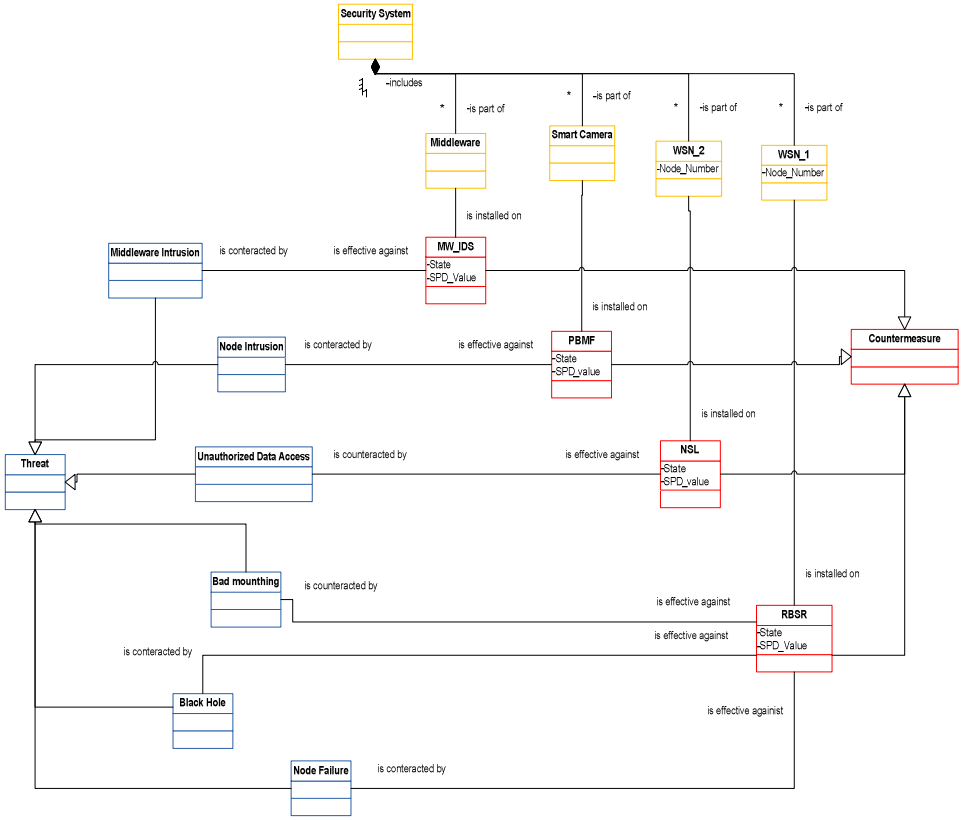
- Middleware SHIELD (MW_SH): the software layer that implements SHIELD methods and mechanisms. In particular, the layer is in charge of performing the discovery and composability activities.
- Reputation-based secure routing (RBSR): each node inside the WSN is equipped with software that enables the selection of a trusted neighbour that guarantees continuity of the routing service. The reputation scheme enables the exchange of first-hand trust evidence used as third-party information by neighbours in building trust relationships.
- Policy-based management framework (PBMF): the SHIELD secure policy-based access control (PBAC) framework facilitates the control of access to devices and their resources via security policies residing on resource-rich infrastructure nodes. It consists of several components that run on different nodes of the nSHIELD architecture. These components are the policy enforcement points (PEP), the policy administration point (PAP), the policy decision points (PDP) and the policy information point (PIP). A node, depending on its capabilities and the available resources, might include one or more of these functional components. The solution adopted for secure PBAC is based on extensible access control markup language (XACML) policies. The PBAC framework is DPWS-compliant, utilising the relevant specifications and existing work to provide message-level security and fine-grained security policy functionality while maintaining interoperability with the standard.
- Network security layer (NSL): this software layer allows a sensor node running Contiki OS to communicate with an external host (e.g., a laptop running Linux), using end-to-end security on the network layer. In this way, any readings from the sensor (e.g., temperature) are transferred via a secure communication channel using IPsec, based on AES-CCM (RFC 4309). Since the sensor communicates via 6LoWPAN and the laptop via standard IPv6, another sensor is used as a bridge between these two technologies. Finally, the security level of this IPsec communication can be changed by modifying the size of the integrity check value (ICV).

- Middleware intrusion detection (MW_IDS): this module protects the middleware entry points from overload or blocked service situations. Overload can be caused either by normal requests due to some bottlenecks or delays in the system, or as a result of a malicious attack, due to a large number of requests or specially crafted requests causing the system to malfunction (DoS or DDoS attacks). The objective of the filtering and intrusion detection functionalities is to provide protection and safe recovery when one of the above types of malformed traffic occurs.

The use of these prototypes makes the Railway Security scenario SHIELD-compliant, providing the SPD-enhancements of the SHIELD methodology. In particular, the aim is to demonstrate that:

- the communication between nodes is secure, due to the presence of mechanisms able to detect cyber-security attacks and encrypt connections
- the communication with the middleware is monitored and protected from possible malicious connections
- stored and transmitted data is protected from unauthorised access
- WSN data routes feature redundant links, with automatic detection of HW/SW node failures and appropriate reconfiguration.

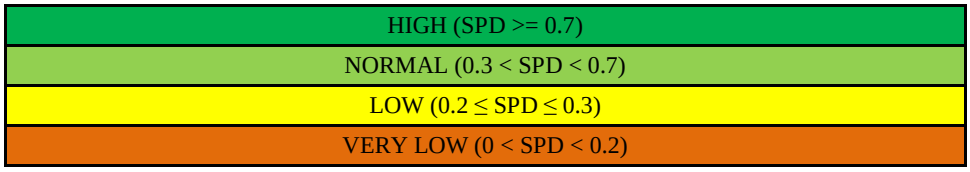
Figure 16 UML model of system-threats-countermeasures (see online version for colours)



In Figure 16, the role of each prototype in the demonstration scenario is defined by means of a UML class diagram.

The MW_IDS prototype is installed at the middleware level and it is able to counteract malicious requests/intrusions against the middleware. The PBMF is installed on specific nodes (in this case on the smart-camera) to detect intrusions at node level. The NSL is a cryptographic protocol installed at the node level, in this case on a WSN, to encrypt the information exchanged between nodes. The RBSR is installed at the node level, in this case on a WSN, to detect malicious threats such as bad mouthing attacks, black hole attacks and node failures, and to restore system operation by reconfiguring the routing among nodes.

Figure 17 Thresholds of SPD level (see online version for colours)



3.3 Scenario description

The aim of the scenario is to show the effectiveness of SHIELD prototypes, middleware and metrics in the specific application, by counteracting cyber-attacks and properly re-configuring the system in case of failures. Figure 18 shows the scenario description using a UML sequence diagram, while Figure 19 reports the UML state diagram representing the system during scenario execution.

In the sequence diagram, the actors involved in the scenario are represented. On each message, a *railway scenario state* (RSS) code is indicated. This code identifies the state of the system. Table 4 shows the steps of scenario and the associated SPD metrics. At each step, one or more components change their status and the related SPD value (red text in Table 4) changes accordingly. For each step, it is possible to identify the state of the system and the state of the single prototypes during scenario execution. Furthermore, the column 'SPD norm' shows a normalised SPD value between 0 (lowest relative SPD) and 1 (highest relative SPD). The colour indicates a qualitative SPD level.

In fact, the SPD levels derived from SHIELD metrics are expressed by plain numbers (e.g., 84.705) since they are the results of mathematical formulas. In order to make the SPD level easier to understand and hence to ease situation awareness for operatora, a normalisation of the SPD level between 0 (lowest relative SPD) and 1 (highest relative SPD) has been performed and reported in the column 'SPD normalised value'.

The formula applied for normalisation is the following:

$$SPD_{norm} = \frac{SPD_{act} - SPD_{\min}}{SPD_{\max} - SPD_{\min}}$$

where *SPD*_min (resp. *SPD*_max) is the minimum (resp. maximum) SPD level, while *SPD_act* is the actual SPD level (i.e., the one computed by the middleware).

The associated reference thresholds are reported in Figure 17.

The system starts from a basic SPD configuration when no threats are detected. In order to save smart-sensor node resources, prototypes are configured with basic SPD functionalities and default SPD levels. The state of prototypes is changed in response to attacks, in order to guarantee adequate SPD levels during system operation.

Figure 18 Scenario UML sequence diagram

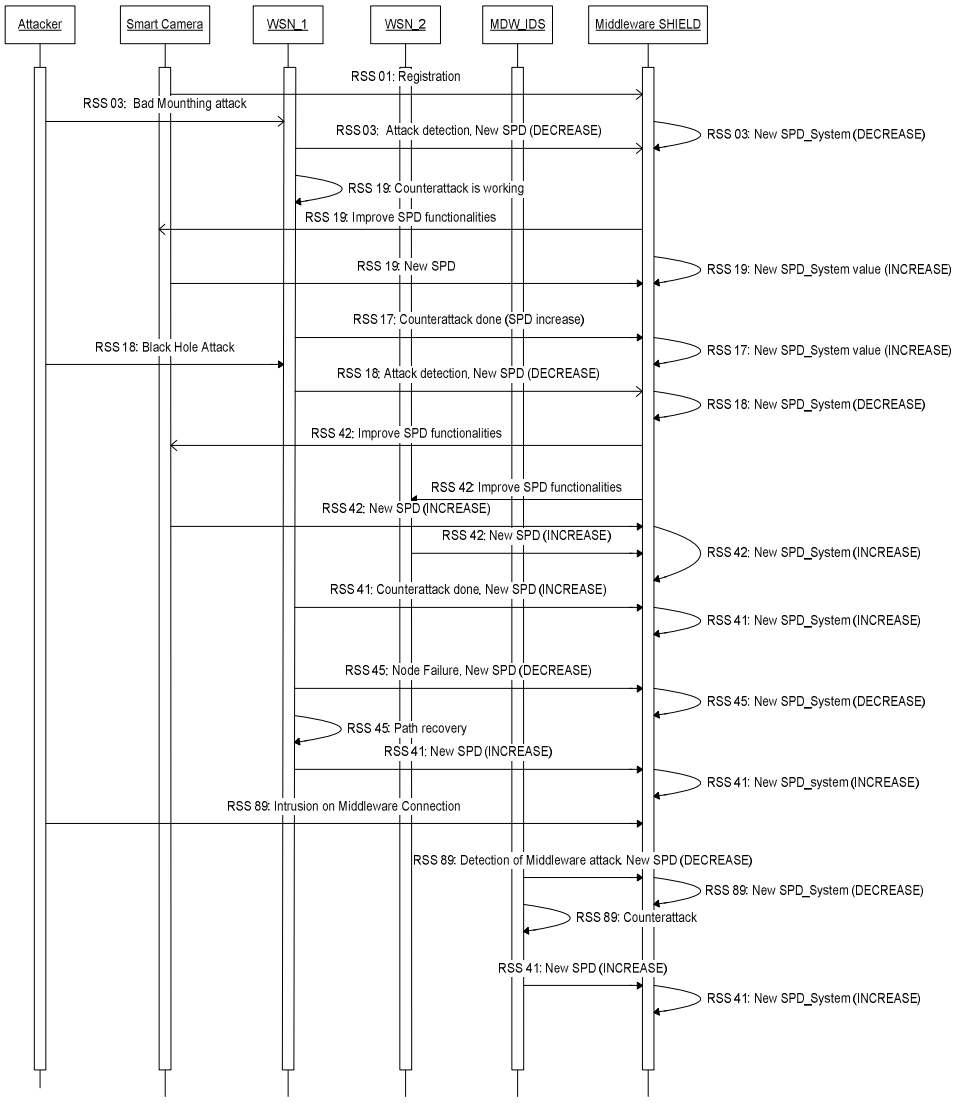


Figure 19 Scenario UML state diagram (see online version for colours)

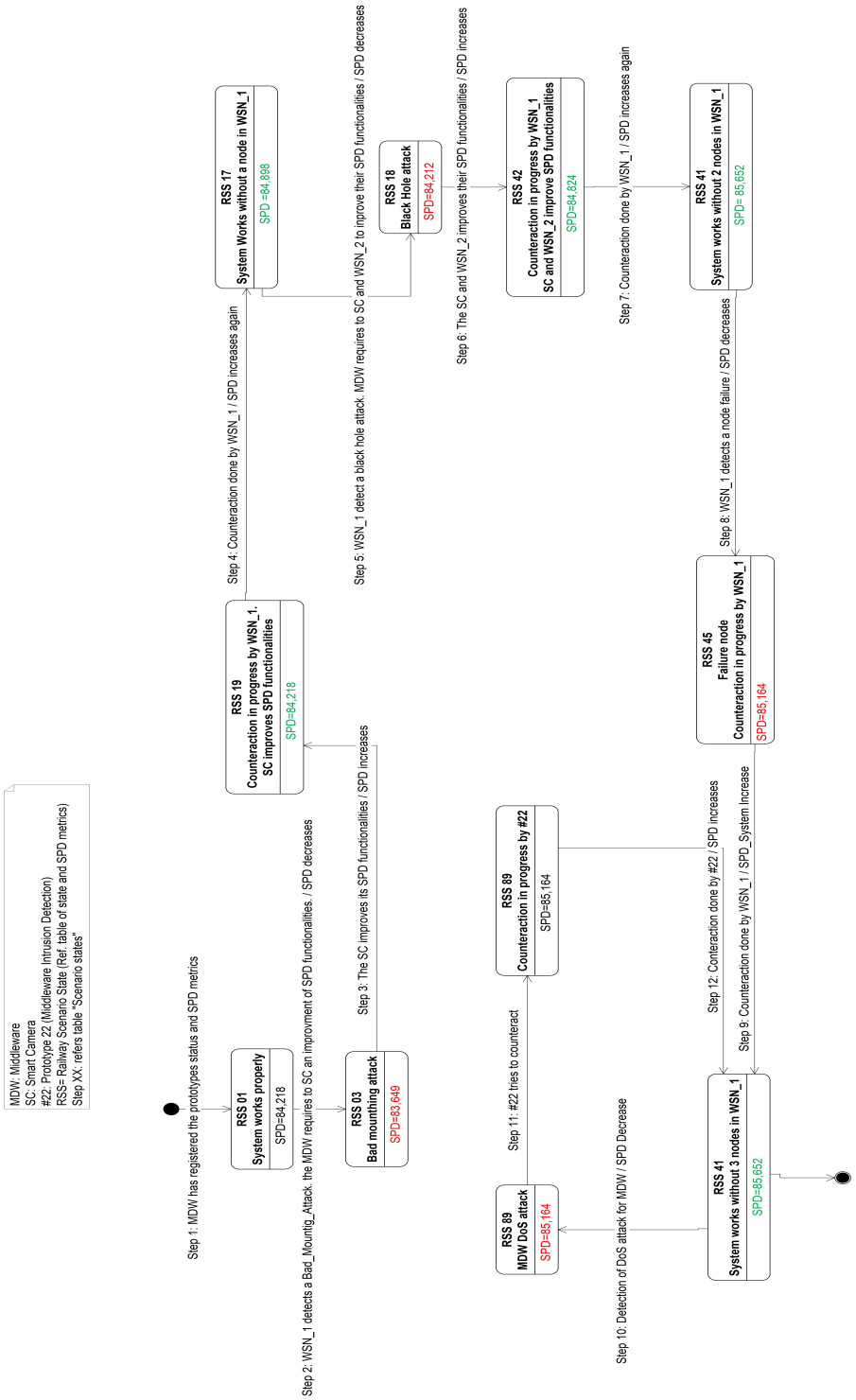


Table 4 Scenario steps (see online version for colours)

Step	*Description*	*RSS*	*SPD norm*
1	Initialisation of all systems and activation of discovery service to register available nodes and prototypes. Basic security functionalities. WSN_1: normal WSN_2: encryption 64 bits Smart camera: messaging – no protection DW_IDS: normal	State_01	0.3
2	In WSN_1, a *bad mouthing attack* is detected. The middleware is informed of the attack is ongoing so it sends a command to the smart camera to activate its security mechanisms. SPD level decreases. WSN_1: bad mouthing attack WSN_2: encryption 64 bits Smart camera: messaging – no protection MDW_IDS: normal	State_03	0
3	Smart-camera activates its SPD functionality. SPD level increases. WSN_1: bad mouthing attack WSN_2: encryption 64 bits Smart camera: messaging – authentication and integrity MDW_IDS: normal	State_19	0.3
4	The WSN_1 has counteracted the bad mouthing attack. SPD level increases again. WSN_1: normal WSN_2: encryption 64 bits Smart camera: messaging – authentication and integrity MDW_IDS: normal	State_17	0.6
5	In WSN_1, a *black hole attack* is detected. The middleware is informed of the attack so it sends a command to the smart-camera and WSN_2 to activate their security mechanisms. SPD level decreases. WSN_1: black hole attack WSN_2: encryption 64 bits Smart camera: messaging – authentication and integrity MDW_IDS: normal	State_18	0.3
6	The smart-camera and WSN_2 have activated their security functionalities. SPD level increases WSN_1: black hole attack WSN_2: encryption 128 bits Smart camera: authentication, integrity and confidentiality MDW_IDS: normal	State_42	0.6

Table 4 Scenario steps (continued) (see online version for colours)

Step	*Description*	*RSS*	*SPD norm*
7	The WSN_1 has counteracted the black hole attack. SPD level increases again. WSN_1: normal WSN_2: encryption 128 bits Smart camera: authentication, integrity and confidentiality MDW_IDS: normal	State_41	1
8	In WSN_1, a *node failure* is detected. The SPD level decreases. WSN_1: dead node alarm WSN_2: encryption 128 bits Smart camera: authentication, integrity and confidentiality MDW_IDS: normal	State_45	0.8
9	WSN_1 recovers from node failure. SPD level increases. WSN_1: normal WSN_2: encryption 128 bits Smart camera: authentication, integrity and confidentiality MDW_IDS: normal	State_41	1
10	A *DoS attack* against middleware is detected. SPD level decreases. WSN_1: normal WSN_2: encryption 128 bits Smart camera: authentication, integrity and confidentiality MDW_IDS: IDS alarm	State_89	0.8
11	End of DoS attack. SPD level increases. WSN_1: normal WSN_2: encryption 128 bits Smart camera: authentication, integrity and confidentiality MDW_IDS: normal	State_41	1

3.4 Demonstration results

The demonstration has proven that the SHIELD system is able to control systemSPD level during the simulated scenario by detecting threats (i.e., attacks and faults) and activating the appropriate countermeasures provided by the installed SHIELD prototypes. That allows fulfilling the customer requirements often referred to as ‘resilience’ or ‘self-healing’.

Figure 20 shows the variation of the SPD Level during scenario execution in the testing environment shown in Figure 21. The SPD level starts with a value of 84.22 and then decreases/increases depending on the events that are happening and on the countermeasures activated by SHIELD, according to the steps described in Table 4.

Figure 20 Variation of the SPD level in the demonstration scenario (see online version for colours)

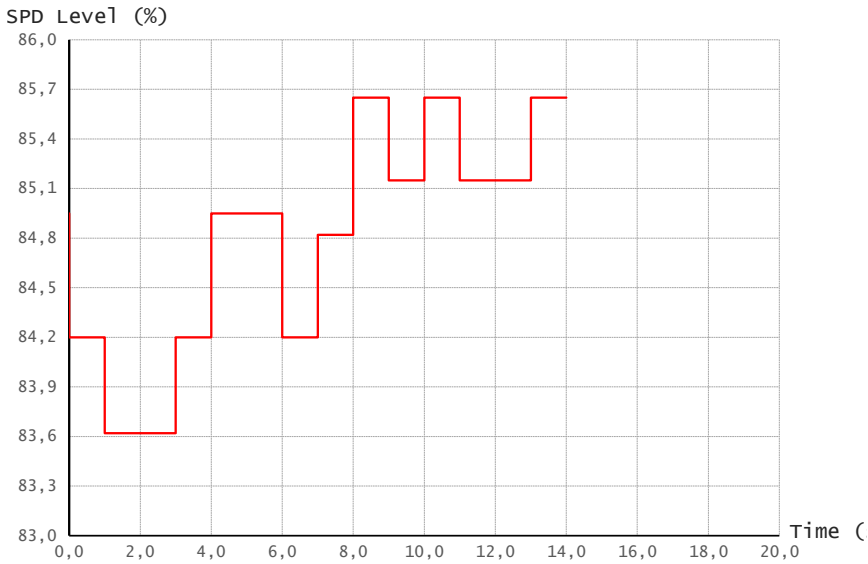
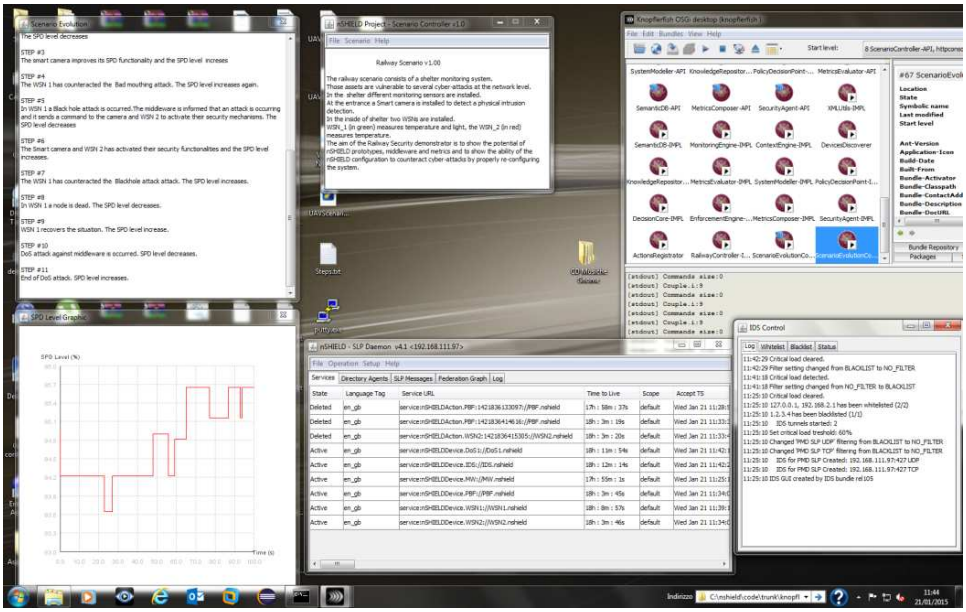


Figure 21 SHIELD testing environment for the railway security scenario (see online version for colours)



The demonstrator validated the correct integration and interoperability among SHIELD components at all levels: middleware and overlay, network and node. The SPD-level output provided in Figure 20 is exactly what was expected: it was the result of SPD-level variations generated by the information passed to the middleware by all

SHIELD components, with decreases caused by some external events, to which the SHIELD system reacts raising the SPD level.

Please note that the final state of the system features a higher SPD-level with respect to the initial one, since more SPD mechanisms have been kept activated in response to the threats detected in the previous steps. Such a 'self-adaptation' to the 'risks' detected in the surrounding environment is something that is highly beneficial to end users, since typical risk assessment is an error prone static activity, that would need to be repeated after some time to redesign the security policies, costing significant time and resources.

Figure 21 shows a screenshot of the PC running the middleware and overlay components, including the secure discovery, the security agent, the intrusion detection module, the OSGi middleware, the semantic model, and the control algorithms. The screenshot has been taken during the final nSHIELD project demonstration held in Nerviano (Selex ES premises) on 21 January 2015.

4 Conclusions

In this paper, we have described the SHIELD approach to ensure cyber-security in railway monitoring and surveillance applications, that are based on increasingly smart embedded devices (environmental sensors and cameras).

The railway security demonstrator has showed a subset of SHIELD functionalities, focusing on the security mechanisms enabled by the involved prototypes. The SHIELD framework has been developed in the context of two EU funded multi-year research projects [namely pilot-SHIELD (pSHIELD Project, http://pshield.unik.no/wiki/PSHIELD-public) and new-SHIELD (pSHIELD Project, http://pshield.unik.no/wiki/PSHIELD-public0]) and it is general enough to address a large number of other possible applications in very different domains. For instance, the new-SHIELD research project demonstration also addressed avionics and social mobility.

We believe the results of the SHIELD project have paved the way to a completely new approach to address the development and control of SPD functionalities, leveraging and integrating the state-of-the-art of current multidisciplinary research in contexts like: hybrid control, semantic modelling, service oriented architectures, computer dependability, critical infrastructure resilience, self-healing and reconfiguration, information security metrics, smart-devices and WSN. Those theoretically outstanding achievements need to be supported by the actual industrialisation of SHIELD-compliant devices that is still in progress. Commercial off the shelf (COTS) SHIELD-compliant devices will allow the SPD-aware composition of heterogeneous devices that will seamlessly integrate to provide dynamic SPD measurement and resilience functionalities.

In railway applications, that allows improving the SPD and shortening the time-to-market of all 'non-vital' (i.e., non-safety-critical) applications, while for the 'vital' ones, a certification process of the framework components will be necessary in order to match the CENELEC requirements for higher safety integrity levels (SIL).

References

Bocchetti, G., Flammini, F., Pappalardo, A. and Pragliola, C. (2009) 'Dependable integrated surveillance systems for the physical security of metro railways', *Proc. 3rd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2009)*, Como, Italy, 30 August to 2 September, pp.1–7.

Bruni, C., Delli Priscoli, F., Koch, G., Palo, A. and Pietrabissa, A. (2016) 'Quality of experience provision in the future internet', *IEEE Systems Journal*, Vol. 10, No. 1, pp.302–312, doi: 10.1109/JSYST.2014.2344658.

Canale, S., Delli Priscoli, F., Di Giorgio, A., Lanna, A., Mercurio, A., Panfili, M. and Pietrabissa, A. (2012) 'Resilient planning of powerline communications networks over medium voltage distribution grids', *Proc. of the 20th Mediterranean Conference on Control and Automation (MED)*, Barcelona, ES, July, pp.710–715 DOI: 10.1109/MED.2012.6265721.

Casola, V., De Benedictis, A., Drago, A., Esposito, M., Flammini, F. and Mazzocca, N. (2012a) 'Securing freight trains for hazardous material transportation: a WSN-based monitoring system', *International Defence and Homeland Security Simulation Workshop (DHSS 2012), in Cooperation with the I3M 2012 Multi-Conference*, Wien, Austria, 19–21 September.

Casola, V., Esposito, M., Mazzocca, N. and Flammini, F. (2012b) 'Freight train monitoring: a case-study for the pSHIELD project', *IEEE 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous*, Palermo, IT, 4–6 July, pp.597–602, DOI: 10.1109/IMIS.2012.51

Common Criteria (2012) *Common Methodology for Information Technology Security Evaluation*, Evaluation Methodology, September, Version 3.1, Revision 4.

Delli Priscoli, F., Fiaschetti, A. and Suraci, V. (2012a) 'The SHIELD framework: how to control security, privacy and dependability in complex systems', *IEEE Workshop on Complexity in Engineering*, pp.1–4, Aachen, June, DOI: 10.1109/CompEng.2012.6242962.

Delli Priscoli, F., Suraci, V., Pietrabissa, A. and Iannone, M. (2012b) 'Modelling quality of experience in future internet networks', *Proc. of the Future Network & Mobile Summit (Future Netw)*, Berlin, DE, 4–6 July, pp.1–9, ISBN: 978-1-905824-16-8.

Di Giorgio, A. and Liberati, F. (2011) 'Interdependency modeling and analysis of critical infrastructures based on dynamic Bayesian networks', *19th Mediterranean Conference on Control and Automation MED11*,Corfu, June, pp.791–797, DOI: 10.1109/MED.2011. 5983016.

ECSEL JU [online] http://www.ecsel-ju.eu/ (accessed July 2016).

Esposito, M., Fiaschetti, A. and Flammini, F. (2013) 'The new shield architectural framework', *ERCIM News*, No. 93, Vol. 93, p.53.

Fiaschetti, A., Lavorato, F., Suraci, V., Palo, A., Taglialatela, A., Morgagni, A., Baldelli, R., Flammini, F. (2011) 'On the use of semantic technologies to model and control security, privacy and dependability in complex systems', in *30th International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2011)*, Springer, pp.467–479, Naples (Italy), 19–22 September, DOI: 10.1007/978-3-642-24270-0_34.

Fiaschetti, A., Morgagni, A., Lanna, A., Panfili, M., Mignanti, S., Cusani, R., Scarano, G., Pietrabissa, A. and Delli Priscoli, F. (2014) 'Control architecture to provide E2E security in interconnected systems: the (new) SHIELD approach', *Proceedings of the 18th International Conference on Circuits, Systems, Communications and Computers (CSCC 2014)*, Santorini Island, 17–19 June, ISBN 978-1-61804-237-8.

Fiaschetti, A., Taglialatela, A., Suraci, V. and DelliPriscoli, F. (2012) 'Semantic technologies to model and control the 'composability' of complex systems: a case study', in *Horizons in Computer Science Research*, Vol. 8, pp.91–110, Nova Science Publisher, ISBN 978-1-62417-413-1.

Flammini, F. (2011) *Critical Infrastructure Security: Assessment, Prevention, Detection, Response*, WIT Press, ISBN 978-1-84564-562-5, Transaction series WIT Transactions on State-of-the-art in Science and Engineering Transaction, Vol. 54.

Flammini, F., Gaglione, A., Ottello, F., Pappalardo, A., Pragliola, C. and Tedesco, A. (2010) 'Towards wireless sensor networks for railway infrastructure monitoring', *Proc. International Conference on Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS'10)*, Bologna, Italy, 19–21 October, pp.1–6.

Hartong, M., Goel, R. and Wijesekera, D. (2008) 'Security and the US rail infrastructure', *International Journal of Critical Infrastructure Protection*, Vol. 1, pp.15–28, No. 1, DOI: 10.1016/j.ijcip.2008.08.006.

Herzog P. (2010) *OSSTMM 3 The Open Source Security Methodology Manual – Contemporary Security Testing and Analysis*, ISECOM [online] http://www.isecom.org/mirror/OSSTMM.3.pdf.

Hodge, V.J., O'Keefe, S., Weeks, M. and Moulds, A. (2015) 'Wireless sensor networks for condition monitoring in the railway industry: a survey', *IEEE Transactions on Intelligent Transportation Systems*, Vol. 16, No. 3, pp.1088–1106.

Manadhata, P.K. and Wing, J.M. (2010) 'An attack surface metric', *IEEE Transactions on Software Engineering*, Vol. 37, No. 3, pp.371–386, DOI: 10.1109/TSE.2010.60.

nSHIELD Project [online] http://www.newshield.eu/ (accessed July 2016).

pSHIELD Project [online] http://pshield.unik.no/wiki/PSHIELD-public (accessed July 2016).

Ramaswamy, S., Fu, H., Sreekantaradhya, M., Dixon, J. and Nygard, K.E. (2003) 'Prevention of cooperative black hole attack in wireless ad hoc networks', *International Conference on Wireless Networks*, June.

Suraci, V., Fiaschetti, A. and Anzidei, G. (2012) 'Design and implementation of a service discovery and composition framework for security, privacy and dependability control', *Future Network & Mobile Summit*, Berlin, DE, July.

Vijaya, K. and Selvam, M. (2013) 'Improving resilience and revocation by mitigating bad mouthing attacks in wireless sensor networks', *International Journal of Scientific & Engineering Research*, April, Vol. 4, No. 4, pp.276–280, 276ISSN 2229-5518.