Definition and Verification of Security Configurations of Cyber-Physical Systems

Ángel Jesús Varela-Vaca[1(✉)], David G. Rosado[2], Luis Enrique Sánchez[2], María Teresa Gómez-López[1], Rafael M. Gasca[1], and Eduardo Fernández-Medina[2]

[1] Dpto. Lenguajes y Sistemas de Informóticos, IDEA Research Group, Universidad de Sevilla, Seville, Spain
{ajvarela,maytegomez,gasca}@us.es

[2] Dpto. Tecnologías y Sistemas de Información, GSyA Research Group, Universidad Castilla-La Mancha, Ciudad Real, Spain
{david.grosado,luise.sanchez,eduardo.fdezmedina}@uclm.es

Abstract. The proliferation of Cyber-Physical Systems (CPSs) is raising serious security challenges. These are complex systems, integrating physical elements into automated networked systems, often containing a variety of devices, such as sensors and actuators, and requiring complex management and data storage. This makes the construction of secure CPSs a challenge, requiring not only an adequate specification of security requirements and needs related to the business domain but also an adaptation and concretion of these requirements to define a security configuration of the CPS where all its components are related. Derived from the complexity of the CPS, their configurations can be incorrect according to the requirements, and must be verified. In this paper, we propose a grammar for specifying business domain security requirements based on the CPS components. This will allow the definition of security requirements that, through a defined security feature model, will result in a configuration of services and security properties of the CPS, whose correctness can be verified. For this last stage, we have created a catalogue of feature models supported by a tool that allows the automatic verification of security configurations. To illustrate the results, the proposal has been applied to automated verification of requirements in a hydroponic system scenario.

Keywords: Cyber-physical system · CPS · Security · Requirement · Feature model · Configuration · Verification

1 Introduction

Cyber-physical systems (CPS) can be defined as systems that collect information from the physical environment via sensors and communication channels, analyse it via controllers and affect the physical environment and relevant processes

via actuators to achieve a specific goal during operation [20]. The use of CPSs facilitates the interaction between the cyberworld and the physical world, but it increases the complexity of the systems derived from the heterogeneity of the CPS components, such as sensors, actuators, embedded systems, controllers, etc.

The correlation between physical and cyber systems brings out new difficulties, that have introduced significant challenges related to security and privacy protection of CPS [15]. In particular, with the complex cyber-physical interactions, threats and vulnerabilities become difficult to assess, and new security issues arise [21], where numerous security threats appear in addition to the traditional cyberattacks [16]. This is the reason why the analysis of the cybersecurity is a key feature in the CPS architecture, to ensure that CPS capabilities are not compromised by malicious agents. Moreover, it is relevant to analyse that the information used (i.e., processed, stored or transferred) has its integrity preserved and the confidentiality is kept where needed [19].

An important lesson should be learned from the way information systems had been engineered in the past is that security often came as an afterthought [13]. If security is not taken into account very early in the development lifecycle, it is nearly impossible to engineer security requirements properly into any complex system. One of the main reasons is that security requirements are often scattered and tangled throughout system functional requirements. Therefore, the security of CPSs should be engineered "by design" early in the development of the CPSs [28,32]. Different studies show that cyber threats have increased in the CPS environment, and there is a need to research how the security requirements can be systematically handled [24,38,42].

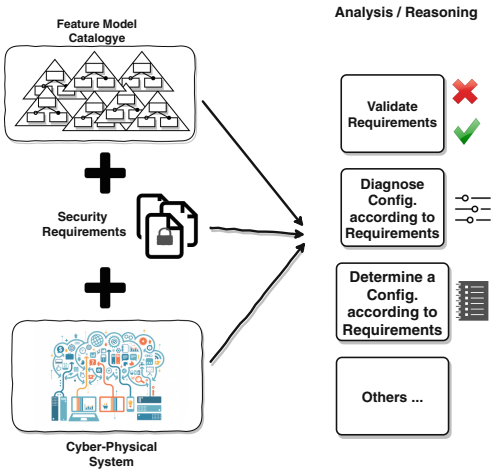

Fig. 1. Overview of the proposal.

The security requirements must include the correct configurations for the CPSs. However, the different types of components, both software and hardware,

involve a high number of possible features that can participate in a CPS. Features about devices, users, platforms, and so on, can provoke a huge number of configurations, both incorrect or correct. Thereby, the description of the security requirements must restrict the incorrect configurations of the features. It implies the analysis of a very high number of possible configurations of the features, to validate if a specific CPS satisfies the defined requirements. Feature models are a well-known technique, belonged to Software Product Lines (SPLs) [8], that provide a mechanism to model and study the satisfiability of the requirements represented by a set of characteristics that can take a set of values restricted by a set of constraints. Derived from the high configurability, and that the features can be shared for various CPSs, we propose the creation of a catalogue of feature models to facilitate the automatic analysis of the security requirements in the context of CPSs. As shown in Fig. 1, the combination of these three elements (cf., Catalogue of Feature Models, Security Requirements and CPSs) will provide a mechanism for reasoning about: the validation of the requirements according to the possible configurations; the diagnosis of misconfigurations, how to ascertain the non-satisfied configurations; the creation of configurations according to the requirements and the feature models, and; other operations such as, in the case of incorrect configurations, the misconfiguration diagnosis by identifying the configuration faults.

To detail the proposal, the paper is organised as follows: Sect. 2 presents an overview of the related work. Section 3 includes a case study of CPS to introduce our proposal. Section 4 tackles the introduction of the main elements of the security requirements for a CPS, using the case study to exemplify the security requirements. Section 5 presents the second part of our proposal, where feature models are introduced as a mechanism to describe the possible correct configurations, that can be stored in a catalogue of Feature Models, and validated automatically concerning the security requirements. Finally, conclusions are drawn and future work is proposed.

2 Related Work

Currently, there is little research associated with software product lines, and security requirements, oriented to cyber-physical systems. Therefore, in this section, some of the main related researches are analyzed.

Related works have been divided into the two areas of research addressed in the article: how feature model analysis have been used in the security and software product lines fields, and; how security requirements and ontologies can be used for the modelling of risk scenarios.

2.1 Cybersecurity and Feature Model Analysis

Feature-Oriented Domain Analysis (FODA) have become mature fields in the Software Product Line (SPL) arena in the last decades [8]. Several are the scenarios where SPLs based on feature model analysis have been applied [18,40],

and different researchers highlight the advantages of these systems since the use of Model-Driven Engineering (MDE) methodology and the Software Product Line (SPL) paradigm is becoming increasingly important [22]. The complexity and the high variability of a CPS, and how SPL can help were analysed in [4,7], detecting the points of variability using feature model analysis. The analysis of the variability of CPS can also support the testing [5].

Security is an understudied field in SPL area. Different approaches have been presented to manage the variability and specify security requirements from the early stages of the product line development [25–27]. Similarly, other approaches addressed the idea of including the security variability into an SPL [36]. In [17], the authors established a software architecture as a reference to develop SPL, dealing with information security aspects. SPLs are currently being targeted for application in CPS, as for some researchers, no standard provides a structured co-engineering process to facilitate the communication between safety and security engineers [11]. For other researches, the information security must be a top priority when engineering C-CPSs as the engineering artefacts represent assets of high value, and the research is focused on the generation of new security requirements stemming from risks introduced by CPSs [10].

On the other hand, there are approaches focused on the security as a use case, such as in [3] and the methodology SecPL [29], where is highlighted the importance of specifying the security requirements and product-line variability. These are annotated in the design model of any system. Other researches developed a security requirements engineering framework for CPSs, that is an extension of SREP [31]. The capacity to support the high variability in the security context though Feature Models appeared in previous papers [23], where the authors study the possible vulnerabilities to create attack scenarios, but not it does not apply to a complex scenario as the CPSs need.

2.2 Ontologies and Security Requirements for Cybersecurity

As seen in the introduction, today's cyber-physical systems require an adequate security configuration. Therefore, some researchers are focusing their research on the development of ontologies and security requirements. Some researchers have developed security tools based on ontologies capable of being integrated with the initial stages of the development process of critical systems, detecting threats and applying the appropriate security requirements to deal with these threats [35]. For other researchers, the use of tools is not enough, since, in this type of system, requirements analysis must consider the details not only of the software but also of the hardware perspective, including sensors and network security. Therefore they propose the development of a security requirements framework for CPSs, analysing the existing ones, and concluding that currently there is no suitable requirement framework for this type of systems. Therefore, they focus on proposing a security requirements engineering framework for CPSs that overcomes the problem of obtaining security requirements for heterogeneous CPS components [33,34]. Other researchers consider that CPSs have unique characteristics that limit the applicability and suitability of traditional

cyber-security techniques and strategies, and therefore propose the development of a methodology of cyber-security requirements oriented to weapons systems [12]. This methodology allows us to discover solutions that improve dimensions (such as security, efficiency, safety, performance, reliability, fault tolerance and extensibility), being possible to use automated coding tools [43]. Additionally, it is also possible to take a more effective approach to understand early the security requirements, during the development of such systems, by using the STPA-Sec [37].

Therefore, we can conclude that at present different researchers have found the need to develop requirement grammars to control the security risks associated with CPSs. Moreover, derived from the complexity of the CPSs, Feature Models have been previously used in the context of the cyber-security.

3 Case Study of a Cyber-Physical System

The case study presented here, which can be seen in Fig. 2, is a CPS system for hydroponic farming, in which different components are involved, both hardware (sensors and actuators) and software (system for storage, monitoring and decision making with Big Data technology).

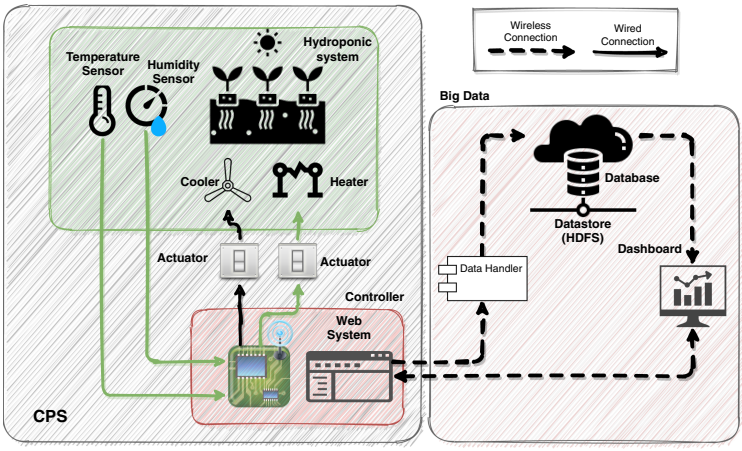

Fig. 2. CPS schema for a hydroponic farming.

The hydroponic farming is controlled by the following physical elements:

- **Temperature and humidity sensors.** They measure the existing temperature and humidity in the environment.
- **Heater and Cooler actuators.** The heater emits heat to increase the ambient temperature and the cooler moves the air to cool the environment. Both actuators are activated or deactivated from the controller.

– **Controller.** It is an Arduino device that receives the data from all the sensors and sends it (via wireless connections) from a web system to the Big Data system.

In addition to the physical part, the controller is connected to a visualisation and control system with Big Data technologies where we have deployed the following components:

– **Dashboard.** It allows the user to control the hydroponic farming in real-time and to consult statistics, as well as to interact (by switching actuators on or off) with the farming, through HTTP requests to the controller.
– **Data handler.** It is responsible for processing the sensor data, received from the controller, and storing it in the database.
– **Datastore.** It contains a Hadoop file system (HDFS) and an HBASE database where all the values coming from the sensors are saved.

4 Security Requirements for Cyber-Physical Systems

CPSs have physical, control and communication requirements, in addition to software security requirements, which make the task of identifying security requirements and translating them into the configuration of our CPS even more complicated. The security requirements represent security features of all types of assets in the system.

Definition 1. ***Security Requirement****. Let SR be a security requirement which consists of a tuple* $\langle AT, SR \rangle$*, where AT is a set of n assets types* $\{at_1, at_2, \cdots, at_n\}$*, and SF is a set of security features* $\{sf_1, sf_2, \cdots, sf_m\}$*.*

The set of assets types (AT) is based on the security recommendations for IoT in the context of critical infrastructures formulated by the ENISA agency [1]. Many of possible security features (SF) for CPS are obtained from OWASP [2] to extract the most important concepts (keys, encryption, protocol, network, AES, SSL, Bluetooth, range, lifetime, etc.). For instance, we can define as a security requirement that the Bluetooth communication between a sensor and a controller is encrypted using the HTTPS protocol, with AES128 encryption and a high confidentiality level. This requirement will generate certain security configurations that must be implemented in the system to ensure compliance with the requirement. This configuration will be verified as valid if the communication, protocol, encryption and confidentiality level defined in the requirement are compatible and correct.

Thereby, the elements that have been considered to define a security requirement (SR) for CPSs are mainly two, “AssetType” (AT) and “SecurityFeature” (SF). The possible values that both AT and SF can take are schematised in Fig. 3, and which are described below:

– **AssetType**: We have classified the types of assets into: “Device”, “User”, “Platform”, “Infrastructure”, “Applications and Services”, and “Information and Data”. Some are divided into other assets as can be seen in Fig. 3.

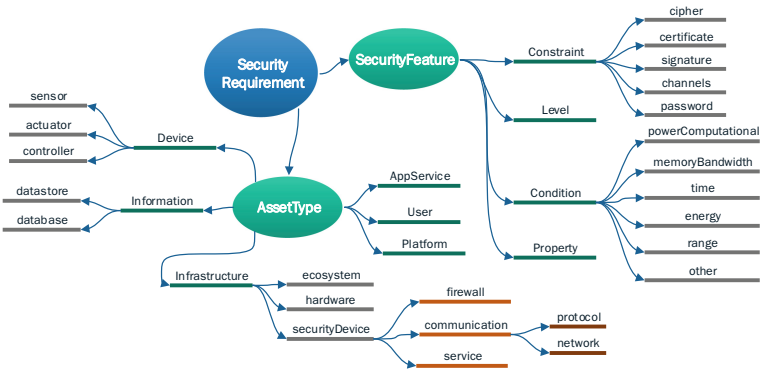

Fig. 3. Elements of a Security Requirement for CPS.

– **SecurityFeature**: This element defines the security features and needs for a security requirement, such as the associated property, the security level, or the conditions and constraints to be taken into account in a CPS.
  - **securityProperty** defines the relationship between a requirement and the security property according to the purpose and context of the requirement. For example, the protection of transmitted information is associated with the property of "Confidentiality" and "Integrity".
  - **securityLevel** indicates the level of security of the requirement and serve to prioritise the requirements during the development, which can range from a high to a low value.
  - **securityConstraint** indicates all possible security-related constraints on the system, such as the strength of passwords, what type of cryptographic or secure communication protocol is used, etc.
  - **securityCondition** indicates the limitations of a CPS that can influence the decision of how to protect the system; for example, if the device has little memory because it will not be able to support certain cryptographic algorithms, or its lifetime to properly define a correct availability service, etc.

4.1 Representation of Security Requirements in JSON

JSON claims to be a useful format for data publication and exchange in many different fields of application and many different purposes. It can be used to exchange information between different technologies, which makes it very useful and attractive to be used to represent the security requirements of a system and to be understood by any language and technology involved. We have proposed a JSON schema to represent the security requirements most easily, and that can be understood by the different applications easily. Part of the syntax of the proposed JSON schema to represent security requirements can be seen in Listings 1.1 and 1.2. This schema represents the properties indicated in Fig. 3. In

Listing 1.1 the schema for "AssetType" is shown, which is an object type with the elements "user", "platform", "device", "infrastructure", "information" and "appservice":

- **User** describes the possible users of the system, which are "consumer", "provider", "process", and "third-party".
- **Platform** includes values of "web-based services" and "Cloud infrastructure and services".
- **Device** contains all the devices of a CPS that are "sensor", "actuator" and "controller".
- **Infrastructure** defines the assets such as "ecosystem", "hardware", "security device" and "communication".
- **Information** determines whether the information is stored in a datastore and/or a database, in transit or in use.
- **AppService** defines all assets related to "analytics and visualization", "device and network management" and "device usage".

Listing 1.1: JSON Schema proposed. Tag: assetType

```
"AssetType": {"type":"object", "properties": {
  "user": {"type":"string",
    "enum": ["consumer","provider","process","third-party"]},
  "platform": {"type":"string",
    "enum": ["web-basedService", "CloudInfrastructure"]},
  "device": {"type":"object","minProperties":1, "maxProperties":3, "properties": {
    "sensor": {"type":"string", "enum": ["humidity","temperature", "accoustic",
            "presure","motion", "chemical","luminosity","flowmeter"]},
    "actuator": {"type":"string", "enum":["hydraulic","mechanical", "electric",
            "pneumatic","magnetic","thermal","TCP/SCP"]},
    "controller":{"type":"string","enum":["microController","microProcessor","FPGA"]}},
  "infrastructure": {"type": "object", "properties": {
    "ecosystem":{"type":"string","enum":["interface","deviceManage","embeddedSystems"]},
    "hardware": {"type": "string", "enum": ["router", "gateway", "powerSupply"] },
    "securityDevice": {"type": "object",   "properties": {
      "service": {"type": "string",
            "enum": ["CloudAuthentication", "AuthenticationSystem","IDS/IPS"]},
      "firewall": {"type": "string",  "enum": ["software", "hardware"]},
      "communication": {"type": "object",   "properties": {
        "protocol": {"type": "string", "enum": ["BLE","RFID","Wifi","ZigBee","ZWave",
                "CoAPP","MQTT","LoRaWAN"]},
        "network": {"type": "string", "enum": ["PAN","WPAN","WAN","VPN","LAN","WLAN"]}},
  "information": {"type": "object",  "properties": {
    "datastore": {"type": "string",  "enum": ["NFS","GPFS","HDFS"]},
    "database": {"type": "string",  "enum": ["SQL","NoSQL","GraphDB"]} },
  "appService": {"type": "string", "enum": ["data analytics and visualization",
          "device and network management", "device usage"]} },
```

In adittion to "AssetType" tag, in Listing 1.2 we can see the "SecurityFeature" tag with the elements '"securityProperty", '"securityLevel", "securityConstraint" and '"securityCondition":

Listing 1.2: JSON Schema proposed. Tag: securityFeature

```
"SecurityFeature": {"type": "object",  "properties": {
    "securityProperty": {"type": "array",  "items": [{"type": "string", "enum":
      ["Identification","Authentication", "Authorization","Confidentiality","Integrity",
      "Non-repudiation","Availability","Privacy","Trust","Audit","Detection"] }],
      "additionalItems": true },
    "securityLevel": {"type": "string",
      "enum": ["Very High","High", "Medium","Low","Very Low"] },
    "securityConstraint": {"type": "object", "properties": {
      "password": {"type": "string", "enum": ["strong", "weak", "multi-factor"]},
      "cipher": {"type": "string", "enum": ["AES128GCM","Camelia","ChaCha20"]},
      "channels": {"type": "string", "enum": ["SSL/TLS", "HTTPS", "Tunneling"]},
      "signature": {"type": "string", "enum": ["SRP","PSK"]},
      "certificate": {"type":"string","enum":["x509","openPGP","openSSL","SAML"]}}},
    "SecurityCondition": {"type": "object", "properties": {
      "powerComputational": {"type": "string", "enum": ["low","medium","high"]},
      "memoryBandwidth": {"type": "string", "enum": ["low","medium","high"]},
      "range": {"type": "string", "enum": ["low","medium","high"]},
      "time": {"type": "string", "enum": ["low","medium","high"]},
      "energy": {"type": "string", "enum": ["low","medium","high"]},
      "other": {"type": "string"}}
        }}}}
"required": ["assetType","securityFeature"]
```

- **securityProperty** includes the values: “AccessControl”, “Audit”, “Authentication”, “Authorization”, “Availability”, “Confidentiality”, “Detection”, “Identification”, “Integrity”, “Non-repudiation”, “Privacy” and “Trust”.
- **securityLevel** considers only a range of values, from a very high level of importance to a very low level of importance.
- **securityConstraint**: the elements are:
  - **password** can take the values: “strong”, “weak”, or “multi factor”.
  - **cipher** describes the type of encryption algorithm, that can be: “Camelia”, “AES128GCM” or “ChaCha20”.
  - **channels** restricts the communication channels: “SSL/TLS”, “HTTPS”, or “Tunneling”.
  - **signature** indicates if the system supports digital signature: “SRP” or “PSK”.
  - **certificate** indicates the formats for the certificates managed: “x509”, “openPGP”, “openSSL”, or “SAML”.
- **securityCondition**: some features for the system, such as powerComputational, memoryBandwidth, range, time, energy, etc. They can take three possible values: “low”, “medium” and “high”.

4.2 Security Requirements for the Case Study

Using the notation we have presented in Sect. 4.1, we show below some examples of security requirements for our case study of the hydroponic farming. To see their expressive capacity, several requirements are defined at different levels of abstraction, which will give rise to different security configurations, which will later be verified with the features model defined in Sect. 5.3.

– **High level.** The wireless communication between the sensors and/or actuators, of the hydroponic farming, and the Arduino system must be encrypted, ensuring confidentiality. This requirement is defined in our JSON schema as is shown in Listing 1.3.

Listing 1.3: High level security requirement in JSON

```
assetType: {
  device: { sensor: "ALL",
            actuator: "ALL",
            controller: "microController" },
  information: {intransit: true}
  infrastructure: {communications:{protocol:["BLE","RFID"],network:"WPAN"}} },
securityFeature: {
  securityProperty: ["Confidentiality"],
  securityLevel: "high",
  securityContraint: {channel: "HTTPS" } }
```

– **Medium level.** The user who wants to visualise the data of the sensors of temperature and humidity of the hydroponic farming from any place must be authorised by the system of authentication. To activate an actuator like the cooler and/or the heater, the user must authenticate with a 2FA system. This requirement is defined in our JSON schema as is shown in Listing 1.4.

Listing 1.4: Medium level security requirement in JSON

```
assetType: {
    user: "consumer",
    device: { sensor: ["temperature","humidity"],
              actuator: "electric",
              controller: "microController"  },
    infrastructure: {
        securityDevice: {
            service: "AuthenticationSystem" },
            communication: { protocol: "Wifi", network: "WAN"} },
    appService: "analytics&visualization" },
securityFeature: {
    securityProperty: ["Authorization", "Authentication"],
    securityLevel: "high",
    securityContraint: { password: "multi-factor"},
    securityCondition: {other: "access authorised"} }
```

– **Low Level.** The short-range sensors of temperature and humidity are connected to an Arduino controller via Bluetooth. The transmitted information acts under the HTTP client/server protocol but the transmitted information must be secured by applying the SSL/TLS cryptographic protocol over HTTP, ensuring confidentiality. This information is stored encrypted in a local webserver with HDFS and HBASE, ensuring integrity. This requirement is defined in our JSON schema as is shown in Listing 1.5.

Listing 1.5: Low level security requirement in JSON

```
assetType: {
    platform: "web-basedService",
    device: { sensor: ["temperature","humidity"],
              controller: "microController" },
    infrastructure: { communication: {

                        protocol: ["BLE","Wifi"],
                        network: "WLAN"    } },
    information: { datastore: "HDFS", database: "NoSQL"    } },
securityFeature:
    { securityProperty: "Confidentiality",
      securityLevel: "high",
      securityContraint: { channels: "HTTPS"}
    },
    { securityProperty: "Integrity",
      securityLevel: "very high",
      securityContraint: { channels: "SSL/TLS", cipher: "AES128GCM"}
    }
}
```

5 Verification of CPS Security Requirements by Using Feature Models

The high variability of the configurations that can be included in the security requirements that involve CPSs, can generate a high number of configurations, whose verification can be very complex. Feature Models (FMs) represent a mechanism that facilitates the representation and treatment of the possible configurations. In this section, we describe how FMs can be used for automatic analysis of the configurations described and how the creation of a catalogue of feature models can be used for verifying the compliance of the security requirements described in the previous section.

5.1 Feature Models

As aforementioned, the use of Feature Models is a broad technique for analysing Feature-Oriented Domain Analysis (FODA) [8] in Software Product Lines (SPLs). Feature Models (FMs) involve a model that defines the features and their relationships.

Definition 2 Feature Model. *Let FM be a feature model which consists of a tuple (F,R), where F is a set of n features $\{f_1, f_2, \cdots, f_n\}$, and R is a set of relations $\{r_1, r_2, \cdots, r_m\}$.*

There are several notations and formalism to define FMs [6], although the most widely used is that proposed by Czarnecki [8], illustrated in Fig. 4. In general, FM diagrams are composed of six types of relations between a parent feature and its child features, although there exist extensions that enable attributes and extra-functionalities for features:

- *Mandatory* relation when child features are required (cf., Root is mandatory sub-feature of A, Root $\leftrightarrow$ A).
- *Optional* relation when child features are optional (cf., Root optional sub-feature of B, Root $\rightarrow$ B).
- *Alternative* relation when one of the sub-features must be selected (i.e., in general $a_1, a_2, \cdots, a_n$ alternative sub-feature of b, $a_1 \vee a_2 \wedge \cdots \wedge a_n \leftrightarrow \bigvee_{i<j}(a_i \vee \cdots \vee a_j)$).
- *Or-relation* when at least one of the sub-features must be selected (i.e., in general $a_1, a_2, \cdots, a_n$ or sub-feature of b, $a_1 \wedge a_2 \wedge \cdots \wedge a_n \leftrightarrow b$, in the figure $C \leftrightarrow C1 \wedge C2$).
- *Require relation*, when a feature requires the existence of other features with non-direct family relation (cf., in the figure A1 $\rightarrow$ B2).
- *Exclude relation*, when a feature excludes the existence of other features with non-direct family relation (cf., in the figure $\neg$(D $\wedge$ E)).
- *Attributes* associated to features, such as B_{Cost} in the example, that is an Integer attribute attached to feature *B*.

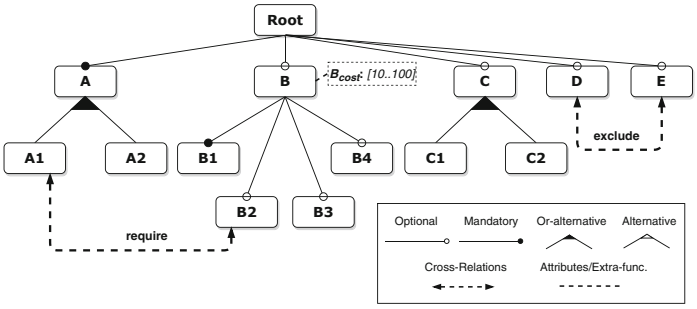

Fig. 4. Toy feature model.

The automated analysis of FMs can be achieved by formal methods [8] based on propositional logic, description logic or constraint programming. Most of the approaches in the literature make a transformation from the FMs to a formalisation, for instance, Constraint Satisfaction Problems (CSPs) or Constraint Optimisation Problems (COPs) [14]. In this work, the tools used to automated the analysis are FaMa and CyberSPL [9,39], both based on the Constraint Programming paradigm.

The automated analysis of FMs enable to perform different reasoning operations on them, for instance, to determine whether the model is valid or not, to obtain the number of all possible configurations, to obtain all possible configurations, even we can ascertain whether it is correct or not concerning the model and based on a configuration. Thus, we can verify a configuration according to the model.

The verification of the security requirement is based on the definition of a valid configuration [39]. Thus, a configuration, c_i, represents an assignment of features for certain FM. For instance, $c_i = \{Root{=}true,\ A = true,\ A1 = true,\ A2 = false, \dots\}$ represents an assignment for the model in Fig. 4, where missed features are assigned to *false* value. The configuration can be represented without the Boolean values but the same semantic, thus, $c_i = \{Root,\ A,\ A1,\ A2, \dots\}$.

The configurations can be *valid* (i.e., correct) whether the selection of assigned features satisfies all the relations, *invalid* otherwise. We revisited the definition of valid configuration [39] to adapt it for the context of the verification of a security requirement (SR) by considering it as a configuration to be checked. Thus, the security requirement represents an assignment according to the features of the model as aforementioned.

Definition 3 Verification of Security Requirements. *Let $\langle FM, SR\rangle$ be the tuple that represents the feature model, FM, and the security requirement, SR, respectively. Let SR be an configuration assignment of n asset type and security features $\{f_1, f_2, \cdots, f_n\}$ according to FM. Thereby, the SR is verified as valid when all the within features of the requirement satisfies the relation of FM.*

$$verify(FM, SR) = \textbf{valid} \iff \{\forall r_i \in FM.R | r_i(SR) \equiv true\} \tag{1}$$

For instance, the configuration assignment $c = \{Root, B\}$ which represent $Root = true$ and $B = true$ is *invalid* due to the relations between B and $B1$ is unsatisfied. This configuration can be seen as a security requirement which represents asset types and security constraints. In our approach, we will formalise a set of FMs that enables the reasoning for the verification of the security requirements presented in previous sections.

5.2 Catalogue of Feature Models for CPS

FM as a formalisation for the definition of security patterns has been also used in [41]. In our approach, FMs are used to formalised the security requirements specified in Sect. 4. To do that, we have formalised a catalogue of FMs that align the security requirements with the recommendation of ENISA [1] for the definition of a security CPS environment. The FMs explained in this section are accessible through the public catalogue of the tool CyberSPL[1].

The FM depicted in Fig. 5 is the result of this synthesis of the ENISA and our proposal for security requirement definition. To bear born in mind, the FM is just an overview since several parts have been hidden for clarity as some require relations and sub-models. As can be seen, the FM is encompassed of two main parts: (1) the assets (cf., Asset) involved in the security requirement, and; (2) the security requirement (cf., Security) specification where properties, conditions, and constraints can be defined.

[1] https://estigia.lsi.us.es/cyberspl/featureModels/publicFeatureModels/.

Regarding the relations, there is a set of requires relations that have been included to explain the relation between asset features and the security requirement aspects. For instance, the data stored into a database may require integrity, hence, the integrity property requires the application of a certain security constraint related to the encryption, e.g., ciphering.

Regarding sub-models, Fig. 6 represents the *Infrastructure* sub-model. The communications at least the specification of used protocols are mandatory but most of the part are optional such as gateways, routers, firewalls, authentication systems, etc.

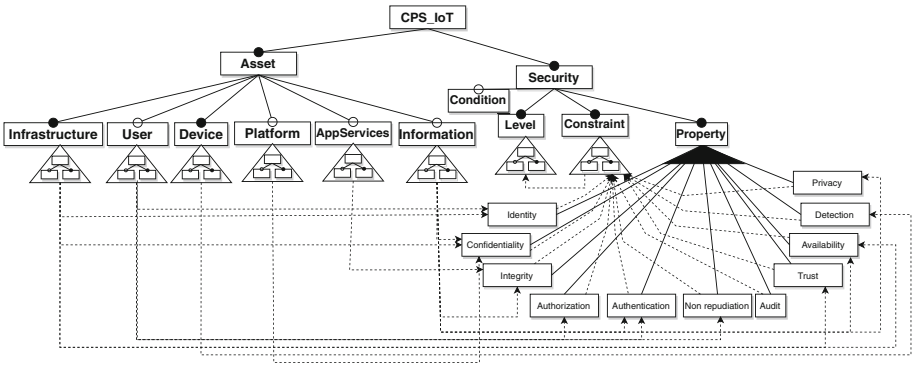

Fig. 5. Feature model for CPS and security requirements.

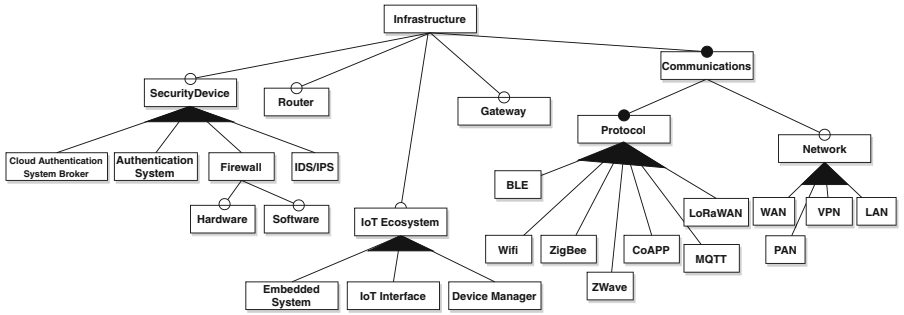

Fig. 6. Sub-model for Infrastructure of CPS.

From the general overview in Fig. 5, we provide different security configuration viewpoints [30] to illustrate some use cases. The first security configuration viewpoint in Fig. 7 is concerning the data-in-transit and confidentiality property. On the one hand, the in-transit data requires a type of network and confidentiality. On the other hand, any communication channel requires confidentiality

properties. The confidentiality can be achieved by the enforcement of the communications using a security protocol such as SSL/TLS. However, the SSL/TLS requires the specification of any cipher methods. To illustrate, we have included three supported by the TLS 1.3 Camelia, AES128GCM, and ChaCha20; these have been matched to three security levels Medium, High, and Very High respectively.

The second security configuration viewpoint in Fig. 8 is concerning data storage and customers. The web-based services can require storage such as databases and users need to be authenticated and authorised to access the sensor data. Thereby, users need authentication and authorisation properties and data storage requires data integrity. On the one hand, integrity can be achieved by the enforcement of cipher methods on the data. These methods Camelia, AES128GCM, and ChaCha20 have been linked to three security levels Medium, High, and Very High respectively. On the other hand, the authentication can require some constraints for the password-based authentication system such as multi-factor. The multi-factor, the length and constraint policy, and the avoid-history based password policy are considered as high level, low level, and a medium level of security respectively.

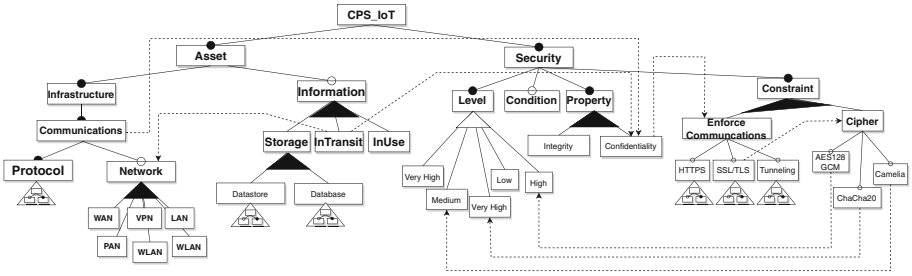

Fig. 7. Feature Model for the security configuration viewpoint of Confidentiality and Data in-transit.

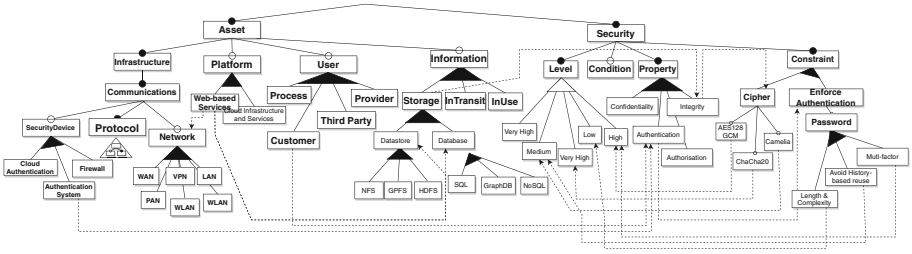

Fig. 8. Feature Model for the security configuration viewpoint of Integrity and Data storage.

To illustrate a running example regarding the security requirements, a security requirement that specifies data-in-transit without protocol and network or a bad combination of SSL/TLS version and the cipher methods concerning the level can provoke an incorrect configuration, therefore, an invalid security requirement. For instance, the data-in-transit through wifi channels requires to the medium level of confidentiality using 3DES or RC4. Both methods have been not considered in our viewpoint because they are deprecated, unrecommended and incompatible with recent versions of the TLS cipher suites.

5.3 Verification Examples for the Case Study

In this section, we show the results for the verification of the three requirements presented in Sect. 3. To do that, we define the configurations for each requirement and subsequently, we verify it against the FMs presented in the previous section.

I. High level Security Requirement

The security requirement establishes that the wireless communication (i.e., WPAN) in transit between the actuators (i.e., ALL), Arduino (i.e., microcontroller), and sensors (i.e., ALL) must be encrypted (i.e., HTTPS), ensuring confidentiality with a high level of security. Based on this specification, we have composed the next security configuration to verify the requirement:

$$\begin{aligned}\mathbf{conf_{HighLevel}} = \{&CPS\_IoT, Asset, Device, Sensor, Humidity, Temperature,\\ &Controller, microController, Actuator, Electric, Magnetic, Infrastructure,\\ &Communications, Protocol, BLE, RFID, Network, WPAN, Information,\\ &Intransit, Security, Enforce\ Communications, Property, Confidentiality,\\ &Level, High, Constraint, HTTPS\} \end{aligned} \quad (2)$$

The $conf_{HighLevel}$ configuration is correctly verified. Thus, it is *valid* since all the features chosen are correct and comply with all the relations in the model. Therefore, we can conclude that *High-Level Security Requirement* of the case study is correct.

II. Medium Level Security Requirement
The security requirement establishes that the users (i.e., Customer) who want to visualise the data (i.e., Data Analytic & Visualisation) of sensors and activate the actuators must be authorised and authenticate (i.e., Authorisation and Authenticate property) by authentication system (i.e., Authentication System) with a 2FA system (i.e., enforce Multi-factor Password).

$$\begin{aligned}\mathbf{conf_{MediumLevel}} = \{&CPS\_IoT, Asset, User, Customer, Device,\\ &Sensor, Humidity, Temperature, Actuator, Electric, Magnetic, Infrastructure,\\ &SecurityDevice, AuthenticationSystem, Communications, Protocol, Wifi,\\ &Network, WAN, AppServices, AnalyticVisualisation, Security, Property,\\ &Authorisation, Authentication, EnforceAuthentication, , Level, High,\\ &Constraint, Password, Multifactor, Condition\} \end{aligned} \quad (3)$$

The $conf_{MedimLevel}$ configuration is verified as *invalid*. The use of communication will require enforcement of the communication to comply with confidentiality properties that are not specified. Therefore, we can conclude that *Medium Level Security Requirement* of the case study is incorrect.

III. Low Level Security Requirement
The security requirement establishes that the sensors connected to the Arduino (i.e, microController) via Bluetooth (i.e., BLE protocol) send information through HTTP protocol but using an SSL/TLS cryptographic protocol to ensure Confidentiality properties. The information is located in encrypted HDFS and HBASE systems to ensure Integrity property. In this case, the requirement specified complementary security properties with two different levels of security, therefore, we need to verify the two configurations one for each security level with the security properties:

$$\begin{aligned}\mathbf{conf}^{\mathbf{High}}_{\mathbf{LowLevel}} = \{&CPS\_IoT, Asset, Device, Sensor, Humidity, Temperature,\\ &Controller, microController, Infrastructure, Communications,\\ &Information, Storage, Datastore, HDFS, Database, NoSQL, Protocol,\\ &BLE, Wifi, Network, WPAN, Security, EnforceCommunications, Property,\\ &Integrity, Confidentiality, Level, High, Constraint, SSLTLS, Cipher,\\ &AES128GCM\} \quad (4)\end{aligned}$$

$$\begin{aligned}\mathbf{conf}^{\mathbf{VeryHigh}}_{\mathbf{LowLevel}} = \{&CPS\_IoT, Asset, Device, Sensor, Humidity, Temperature,\\ &Controller, microController, Infrastructure, Communications,\\ &Information, Storage, Datastore, HDFS, Database, NoSQL, Protocol,\\ &BLE, Wifi, Network, WPAN, Security, EnforceCommunications, Property,\\ &Integrity, Confidentiality, Level, VeryHigh, Constraint, SSLTLS, Cipher,\\ &AES128GCM\} \quad (5)\end{aligned}$$

The $conf^{High}_{LowLevel}$ configuration is verified as *valid* but the $conf^{VeryHigh}_{LowLevel}$ configuration is verified as *invalid* due to the cipher chosen. The AES128GCM cipher method is unsupported for the very high level of security. Thereby, we can conclude that the *Low Level Security Requirement* is incorrect.

Summarising, we demonstrate the reasoning capabilities of the model by verifying the security requirement in which two of the three, i.e, Medium and Low have been verified as *invalid* due to problems in the specification, and just one, i.e, the High level is verified as *valid*.

6 Conclusion and Future Work

The high features that can be configurable in a CPS make difficult the evaluation of the requirements that involve security aspects. To facilitate the validation of the security requirements for CPSs, we propose the use of Feature Models to support the description of the possible configuration. To formalise the security requirement description, we have defined a common grammar to define security requirements for CPSs by using JSON

that gathering the possible involved elements. For an automatic verification of the requirements, Feature Models are used to validate a configuration according to the requirements. Moreover, a catalogue of FMs for CPS has been created and stored in a special repository to be reused for any set of requirement to be validated. The feasibility of the solution, both the description capacity of the requirements and the catalogue of configurations in CPS has been evaluated through a case study of a hydroponic farming CPS. For the future, we plan to extend the types of reasoning that can be applied over the combination of the feature models and the security requirements for CPSs, such as the diagnosis of the configurations that do not satisfy the requirement, or the generation of correct configurations according to a set of requirements specified.

Acknowledgement. This research is partially supported by Ministry of Science and Technology of Spain with projects ECLIPSE (RTI2018-094283-B-C33), by Junta de Andalucía with METAMORFOSIS projects, and Junta de Comunidades de Castilla-La Mancha with the GENESIS project (SBPLY-17-180501-000202); and by European Regional Development Fund (ERDF/FEDER).

References

1. Baseline security recommendations for IoT (2018). https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot
2. OWASP Top Ten. Available from OWASP (2020). https://owasp.org/www-project-top-ten/
3. Arciniegas, J.L., Dueñas, J.C., Ruiz, J.L., Cerón, R., Bermejo, J., Oltra, M.A.: Architecture reasoning for supporting product line evolution: an example on security. In: Kakola, T., Duenas, J.C. (eds.) Software Product Lines, pp. 327–372. Springer, Heidelberg (2006). https://doi.org/10.1007/978-3-540-33253-4_9
4. Arrieta, A., Sagardui, G., Etxeberria, L.: Cyber-physical systems product lines: variability analysis and challenges (2015)
5. Arrieta, A., Wang, S., Sagardui, G., Etxeberria, L.: Search-based test case selection of cyber-physical system product lines for simulation-based validation. In: Mei, H. (ed.) Proceedings of the 20th International Systems and Software Product Line Conference, SPLC 2016, Beijing, China, 16–23 September 2016, pp. 297–306. ACM (2016). https://doi.org/10.1145/2934466.2946046
6. Batory, D.: Feature models, grammars, and propositional formulas. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 7–20. Springer, Heidelberg (2005). https://doi.org/10.1007/11554844_3
7. Beek, M.H.T., Fantechi, A., Gnesi, S.: Product line models of large cyber-physical systems: the case of ertms/etcs. In: Proceedings of the 22nd International Systems and Software Product Line Conference, SPLC '18, vol. 1, pp. 208–214. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3233027.3233046
8. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: a literature review. Inf. Syst. **35**(6), 615–636 (2010). https://doi.org/10.1016/j.is.2010.01.001
9. Benavides, D., Segura, S., Trinidad, P., Cortés, A.R.: Fama: tooling a framework for the automated analysis of feature models. VaMoS **2007**, 01 (2007)

10. Biffl, S., Eckhart, M., Lüder, A., Weippl, E.: Introduction to security and quality improvement in complex cyber-physical systems engineering. Security and Quality in Cyber-Physical Systems Engineering, pp. 1–29. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25312-7_1
11. Bramberger, R., Martin, H., Gallina, B., Schmittner, C.: Co-engineering of safety and security life cycles for engineering of automotive systems. ACM SIGAda Ada Lett. **39**(2), 41–48 (2020)
12. Carter, B., Adams, S., Bakirtzis, G., Sherburne, T., Beling, P., Horowitz, B., Fleming, C.: A preliminary design-phase security methodology for cyber-physical systems. Systems **7**(2), 21 (2019)
13. Cysneiros, L.M., Leite, J.C.S.D.P.: Nonfunctional requirements: from elicitation to conceptual models. IEEE Trans. Softw. Eng. **30**(5), 328–350 (2004). https://doi.org/10.1109/TSE.2004.10
14. Dechter, R.: Constraint Processing. Morgan Kaufmann Publishers Inc, San Francisco (2003)
15. Ding, J.: Intrusion detection, prevention, and response system (IDPRS) for cyber-physical systems (CPSs). In: Securing Cyber-Physical Systems, pp. 371–392. CRC Press, Boca Raton (2015). https://doi.org/10.1201/b19311-16
16. Dorbala, S., Bhadoria, R.: Analysis for security attacks in cyber-physical systems. In: Cyber-Physical Systems, pp. 395–414. Chapman and Hall/CRC, Baco Raton (2015). https://doi.org/10.1201/b19206-23
17. Fægri, T.E., Hallsteinsen, S.: A software product line reference architecture for security. In: Kakola, T., Duenas, J.C. (eds.) Software Product Lines, pp. 275–326. Springer, Heidelberg (2006). https://doi.org/10.1007/978-3-540-33253-4_8
18. Galindo, J.A., Benavides, D., Trinidad, P., Gutiérrez-Fernández, A.-M., Ruiz-Cortés, A.: Automated analysis of feature models: Quo vadis? Computing **101**(5), 387–433 (2018). https://doi.org/10.1007/s00607-018-0646-1
19. Griffor, E., Wollman, D., Greer, C.: Framework for Cyber-Physical Systems: Volume 1, Overview. Technical Report, June, National Institute of Standards and Technology, Gaithersburg, MD (2017). https://doi.org/10.6028/NIST.SP.1500-201
20. Gunes, V., Peter, S., Givargis, T., Vahid, F.: A survey on concepts, applications, and challenges in cyber-physical systems. KSII Trans. Internet Inf. Syst. **8**(12), 4242–4268 (2014). https://doi.org/10.3837/tiis.2014.12.001
21. Humayed, A., Lin, J., Li, F., Luo, B.: Cyber-physical systems security - a survey. IEEE Internet Things J. **4**(6), 1802–1831 (2017). https://doi.org/10.1109/JIOT.2017.2703172
22. Iglesias, A., Iglesias-Urkia, M., López-Davalillo, B., Charramendieta, S., Urbieta, A.: Trilateral: software product line based multidomain IoT artifact generation for industrial CPS. In: Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, vol. 1, pp. 64–73. SCITEPRESS-Science and Technology Publications, Lda (2019)
23. Kenner, A., Dassow, S., Lausberger, C., Krüger, J., Leich, T.: Using variability modeling to support security evaluations: virtualizing the right attack scenarios. In: VaMoS '20: 14th International Working Conference on Variability Modelling of Software-Intensive Systems, Magdeburg, Germany, 5–7 February 2020, pp. 10:1–10:9 (2020). https://doi.org/10.1145/3377024.3377026
24. Liu, Y., Peng, Y., Wang, B., Yao, S., Liu, Z.: Review on cyber-physical systems. IEEE/CAA J. Automatica Sinica **4**(1), 27–40 (2017). https://doi.org/10.1109/JAS.2017.7510349

25. Mellado, D., Fernández-Medina, E., Piattini, M.: Security requirements management in software product line engineering. In: Filipe, J., Obaidat, M.S. (eds.) ICETE 2008. CCIS, vol. 48, pp. 250–263. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05197-5_18
26. Mellado, D., Fernández-Medina, E., Piattini, M.: Towards security requirements management for software product lines: a security domain requirements engineering process. Comput. Stand. Interfaces **30**(6), 361–371 (2008)
27. Mellado, D., Mouratidis, H., Fernández-Medina, E.: Secure tropos framework for software product lines requirements engineering. Comput. Stand. Interfaces **36**(4), 711–722 (2014)
28. Nguyen, P.H., Ali, S., Yue, T.: Model-based security engineering for cyber-physical systems: a systematic mapping study (2017). https://doi.org/10.1016/j.infsof.2016.11.004
29. Peldszus, S., Strüber, D., Jürjens, J.: Model-based security analysis of feature-oriented software product lines. In: Proceedings of the 17th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, pp. 93–106 (2018)
30. Publishing, V.H.: The TOGAF Standard, Version 9.2. TOGAF series, Van Haren Publishing (2018). https://books.google.es/books?id=XQ6DtgEACAAJ
31. ur Rehman, S., Allgaier, C., Gruhn, V.: Security requirements engineering: a framework for cyber-physical systems. In: 2018 International Conference on Frontiers of Information Technology (FIT), pp. 315–320. IEEE (2018)
32. Rehman, S., Gruhn, V.: An effective security requirements engineering framework for cyber-physical systems. Technologies **6**(3), 65 (2018). https://doi.org/10.3390/technologies6030065
33. Rehman, S., Gruhn, V., Shafiq, S., Inayat, I.: A systematic mapping study on security requirements engineering frameworks for cyber-physical systems. In: Wang, G., Chen, J., Yang, L.T. (eds.) SpaCCS 2018. LNCS, vol. 11342, pp. 428–442. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05345-1_37
34. Rehman, S.U., Gruhn, V.: An effective security requirements engineering framework for cyber-physical systems. Technologies **6**(3), 65 (2018)
35. Shaaban, A.M., Gruber, T., Schmittner, C.: Ontology-based security tool for critical cyber-physical systems. In: Proceedings of the 23rd International Systems and Software Product Line Conference, vol. B, pp. 207–210 (2019)
36. Sion, L., Van Landuyt, D., Yskout, K., Joosen, W.: Towards systematically addressing security variability in software product lines. In: Proceedings of the 20th International Systems and Software Product Line Conference, pp. 342–343 (2016)
37. Span, M., Mailloux, L.O., Mills, R.F., Young, W.: Conceptual systems security requirements analysis: aerial refueling case study. IEEE Access **6**, 46668–46682 (2018)
38. Subramanian, N., Zalewski, J.: Quantitative assessment of safety and security of system architectures for cyberphysical systems using the NFR approach. IEEE Syst. J. **10**(2), 397–409 (2016). https://doi.org/10.1109/JSYST.2013.2294628
39. Varela-Vaca, A.J., Gasca, R.M., Ceballos, R., Gómez-López, M.T., Bernáldez Torres, P.: CyberSPL: a framework for the verification of cybersecurity policy compliance of system configurations using software product lines. Appl. Sci. **9**(24) (2019). https://doi.org/10.3390/app9245364

40. Varela-Vaca, Á.J., Galindo, J.A., Ramos-Gutiérrez, B., Gómez-López, M.T., Benavides, D.: Process mining to unleash variability management: discovering configuration workflows using logs. In: Proceedings of the 23rd International Systems and Software Product Line Conference, vol. A, pp. 265–276 (2019)
41. Varela-Vaca, Á.J., Gasca, R.M.: Formalization of security patterns as a means to infer security controls in business processes. Logic J. IGPL **23**(1), 57–72 (2015). https://doi.org/10.1093/jigpal/jzu042
42. Yoo, H., Shon, T.: Challenges and research directions for heterogeneous cyber-physical system based on IEC 61850: Vulnerabilities, security requirements, and security architecture. Fut. Gener. Comput. Syst. **61**, 128–136 (2016). https://doi.org/10.1016/j.future.2015.09.026
43. Zhu, Q., Sangiovanni-Vincentelli, A.: Codesign methodologies and tools for cyber-physical systems. Proc. IEEE **106**(9), 1484–1500 (2018)