Integrating Security into the Product-Line-Engineering Framework: A Security-Engineering Extension

Christian Biermann[1,2], Richard May[1] and Thomas Leich[1]
[1]*Harz University of Applied Sciences, Wernigerode, Germany*
[2]*Msg Systems ag, München, Germany*

Keywords: Product-Line Engineering, Security Engineering, Security, Business Processes, Framework.

Abstract: Modern software systems are becoming increasingly configurable, often relying on Product-Line Engineering (PLE) to efficiently develop variant-rich systems while ensuring reusability. However, security considerations in existing PLE research are typically insufficient as security is often (partly) neglected or not integrated into the overall development process. To address this gap, we developed an additional layer of the PLE framework: security engineering — positioned between domain engineering and application engineering. Our results are based on a systematic review of 49 secure PLE frameworks and workflows, synthesizing their insights and our expertise in compliance with the ISO/IEC 27000 series. By following six processes and 12 activities, our iterative approach ensures that security is systematically embedded in the PLE process. We particularly highlight the importance of reusable security artifacts, secure business-process modeling, and standard compliance, aiming to facilitate the transfer of theoretical solutions into secure business practice.

1 INTRODUCTION

Modern software systems are becoming increasingly configurable, i.e., they are based on a variety of features which can be disabled, enabled, or combined (Abal et al., 2018). In this context, they often build on concepts and techniques related to Product-Line Engineering (PLE) as an established approach for developing and maintaining configurable software systems. In particular, PLE supports the creation of product families with similar but adapted functionalities, optimizing development efficiency (i.e., reusability, maintenance) and reducing development time and costs (Pohl et al., 2005; Apel et al., 2013). This approach has already shown its benefits in various domains, for example, manufacturing (Iglesias et al., 2017), healthcare (Gomes et al., 2012), or Enterprise-Resource-Planning (ERP) (May et al., 2023a).

In recent years, configurable systems have become increasingly complex, involving a wide diversity of data, features, business processes, as well as associated interactions, configurations, and dependencies (Mellado et al., 2014; May et al., 2023a). However, due to this complexity, the potential to fall victim to a cyberattack is constantly increasing (Abal et al., 2018). More specifically, the more complex (i.e., configurable) the systems are, the greater the risk of unexpected vulnerabilities and their exploitation by cyberattacks due to potential misconfigurations, unintended features interactions, or unknown dependencies. Therefore, increasing variability leads to major challenges related to IT security (Mesa et al., 2018; Kenner et al., 2021).

This is why current research already considers parts of secure PLE, such as feature verification, or broader PLE workflows, such as requirements-engineering frameworks (cf. Section 6). However, recent studies (Kenner et al., 2021; May et al., 2022, 2023b, 2024a) have already identified a lack of (holistic) security considerations in PLE, i.e., security is typically only referred to as a non-functional requirement or quality objective rather than a cross-cutting concern equally important as other functional requirements. This treatment usually leads to an under-representation of security in the PLE processes and activities. Although some frameworks already consider security on a somewhat equal level with other functional requirements and associated features, they usually only focus on a few parts of the PLE process, such as security verification with CyberSPL by Varela-Vaca et al. (2019). In addition, current solutions often do not comply with security standards, such as ISO/IEC 27000 (2022), impairing the transfer of theoretical approaches into business practice. This

Biermann, C., May, R. and Leich, T.
Integrating Security into the Product-Line-Engineering Framework: A Security-Engineering Extension.
DOI: 10.5220/0013489500003964
In *Proceedings of the 20th International Conference on Software Technologies (ICSOFT 2025)*, pages 75-86
ISBN: 978-989-758-757-3; ISSN: 2184-2833

finding does not only refer to certain domains, instead it represents a critical cross-domain gap (e.g., in manufacturing).

To address this gap, we developed an extension of the PLE framework (Pohl et al., 2005) by analyzing and synthesizing 49 existing frameworks and workflows (Nickerson et al., 2013) supporting secure PLE. In particular, we ensured that the entire extension complies with the requirements of the ISO/IEC 27000 (2022) series. As a result, we propose an additional PLE layer between domain engineering and application engineering, called *security engineering*. We argue that the extension is currently domain-independent, but may be easily adaptable to certain domains by considering domain-specific standards and regulations in the problem space (cf. Section 4). Overall, we contribute the following:

- An overview of current research regarding security in the context of PLE frameworks.
- An additional, standard-compliant layer of PLE, called *security engineering*, comprising six processes and 12 activities.
- An open-access repository including the extraction sheet to allow study replications.[1]

With our work, we aim to create a broader understanding of how security concerns can be holistically considered in PLE. Moreover, we believe that security engineering can support both researchers and practitioners regarding the transfer of theoretical knowledge into business practice, ensuring compliance with security standards. However, note that the PLE framework including our extension is not yet validated, although it is based on peer-reviewed information in most parts.

In the remainder of this paper, we organize the content as follows: First, Section 2 provides an overview of the relevant background. After that, in Section 3, we detail our methodology and present a summary of our development steps. Section 4 introduces the our extension of the PLE framework. In Section 5, we discuss further implications and address potential threats to the internal and external validity. Then, in Section 6, we review related work, and finally, Section 7 concludes the paper.

2 BACKGROUND

Next, essential background information on PLE and security management are provided.

2.1 Product-Line Engineering

PLE is an established methodology for developing software-product families that share common characteristics and features, built on a shared core architecture and configured for specific needs (Pohl et al., 2005; Thüm et al., 2014). Its primary goals are to optimize software development by improving efficiency and reusability, reducing costs, and accelerating development (van der Linden et al., 2007). A key aspect of PLE is managing software variability to tailor products to specific requirements. Feature models are used to represent and configure both static and dynamic variability, enabling systematic reuse of software artifacts and automated product derivation (Batory, 2004; Apel et al., 2013). However, PLE introduces complexity, requiring specialized strategies to verify system correctness. Feature-based, product-based, and family-based analyses address variability challenges in different domains (Thüm et al., 2014). Security requirements, often cross-cutting functional variability, require holistic strategies that integrate security into both domain and application engineering to ensure the quality, reliability, and security of derived products (Horcas et al., 2019; May et al., 2025).

2.2 Security Management

Security management is a business-critical component of software systems, requiring a holistic approach to protect their assets. At its core lies the CIA triad, encompassing the security goals of confidentiality, integrity, and availability (Samonas and Coss, 2014). These principles form the foundation of widely recognized standards such as the ISO/IEC 27000 (2022) guiding organizations in achieving robust security in practice. IT security focuses on protecting software systems and their associated assets from threats, for example, including breaches, unauthorized access, or data manipulation (NIST SP 800-30r1, 2012; ISO/IEC 27000, 2022). Threats are typically identified by monitoring systems and vulnerability databases (NIST SP 800-150, 2016), while risks are related to their potential impact and likelihood (ISO/IEC 27000, 2022). Vulnerabilities, i.e., weaknesses exploitable by attacks or system failures, are key to understanding risks. Effective risk management integrates security activities across all phases of the software-development lifecycle to meet security requirements and ensure an acceptable level of risk (NIST SP 800-30r1, 2012).

[1]https://doi.org/10.5281/zenodo.14748327

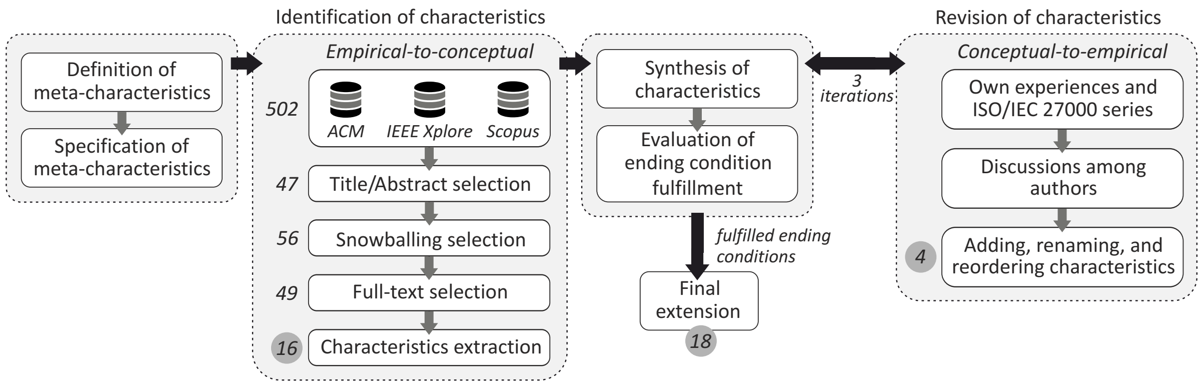
Identification of characteristics
Revision of characteristics
Definition of meta-characteristics
Specification of meta-characteristics
Empirical-to-conceptual
502
ACM
IEEE Xplore
Scopus
47
Title/Abstract selection
56
Snowballing selection
49
Full-text selection
16
Characteristics extraction
Synthesis of characteristics
Evaluation of ending condition fulfillment
3 iterations
fulfilled ending conditions
Final extension
18
Conceptual-to-empirical
Own experiences and ISO/IEC 27000 series
Discussions among authors
4
Adding, renaming, and reordering characteristics

Figure 1: Method overview; numbers indicate amount of papers or characteristics (dark circled), i.e., processes and activities.

3 METHOD

In the following section, we describe the method of our study (i.e., design and conduct).

3.1 Design

The framework extension builds on the methodology proposed by Nickerson et al. (2013), which offers guidelines for developing frameworks, making it particularly well-suited to our context. The method involved three main steps:

Meta-Characteristics. First, meta-characteristics were defined to capture engineering processes and activities concerning IT security and PLE. We focus exclusively on security engineering as a third perspective, complementing domain and application engineering. Security engineering builds on domain engineering results and provides requirements and features for application engineering, ensuring equal consideration of functional and secure development. We distinguish three perspectives: the problem space (i.e., domain-specific abstractions such as requirements), the solution space (i.e., realization of these abstractions), and the mapping (i.e., configuration options to adapt abstractions to specific contexts) (Berg et al., 2005).

Ending Conditions. Second, ending conditions (E_i) ensure the methodological process maintains high quality (Nickerson et al., 2013). Eight conditions were considered: four objective criteria ensure unique characteristics (E_1), original context is preserved (E_2), no merging in the last iteration (E_3), and no additions in the last iteration (E_4). Four subjective criteria ensure robustness (E_5), completeness (E_6), adaptability (E_7), and explainability (E_8).

Characteristics Development. Third, characteristics were developed iteratively using an empirical-to-conceptual approach (Nickerson et al., 2013), supported by a systematic literature review (Kitchenham et al., 2015). We focused on secure PLE frameworks and workflows rather than specific engineering processes. The following search string was formulated:

```
 ("secur*") AND ("framework" OR "taxonomy"
OR "engineering" OR "pattern" OR "process"
OR "approach" OR "architecture") AND
("product line")
```

Inclusion/exclusion criteria limited research to peer-reviewed work (2005–2024) with at least four pages, excluding studies not focused on secure PLE frameworks or workflows. On November 1, 2024, the second author conducted an automated search across ACM GUIDE TO COMPUTING LITERATURE (21 articles), IEEE XPLORE (100 articles), and SCOPUS (381 articles), yielding 502 results. After title/abstract screening, 47 papers remained. Two iterations of forward and backward snowballing added nine more papers. After full-text review, 49 papers were included (cf. Figure 1 and Table 1).

The selected works were analyzed using the defined meta-characteristics, and processes and activities were abstracted and synthesized. The first and second authors discussed these results, evaluating them against the ending conditions. All conditions except E_1, E_3, E_4, E_5, and E_6 were met. This iteration produced the first extension version, including 14 characteristics (six processes, eight activities).

Subsequent iterations used a conceptual-to-empirical approach (Nickerson et al., 2013), relying on the authors' expertise and the ISO/IEC 27000 (2022) standards. Four activities related to security testing were iteratively added, reordered, and renamed. Each iteration included discussions using an open-card sorting method and evaluations of ending condition fulfillment. After three iterations, the final extension consisted of six processes and 12 activities (cf. Figure 1 and Figure 2).

4 SECURITY ENGINEERING

To address security concerns throughout the PLE development, we present our framework extension: *security engineering* (cf. Figure 2). The six identified processes and 12 activities, classified into the three key perspectives of problem space, mapping, and solution space are described in detail below.

4.1 Problem Space

The problem space focuses on understanding (i.e., analyzing) and defining the security landscape as well as the architecture. This perspective is crucial since it sets the foundation for all subsequent security efforts.

Security Analysis. The security analysis is the first step in addressing security concerns in product lines. It involves a detailed examination of the domain to identify legal requirements, stakeholder expectations, potential risks, and threats.

One of the activities in this process is the analysis of *domain legal regulations*. This involves understanding the legal and regulatory frameworks that apply to more general domains, such as the ISO/IEC 27000 (2022) series for information security, or specific domains, such as the Health Insurance Portability and Accountability Act (Fernandez et al., 2015) in healthcare. Here, it is highly important to identify relevant guidelines and restrictions to align the overall security concept with them, ensuring compliance with legal obligations.

Another activity in security analysis is the *domain requirements* identification, which is similar to the procedure in the traditional domain engineering (Pohl et al., 2005; Reinhartz-Berger et al., 2013), however, focusing on security. This step includes capturing the specific security needs of stakeholders involved in the development process of a specific system, such as requirements linked to assets and features of cyber-physical systems (Varela-Vaca et al., 2021).

Closely connected to the domain requirements, *domain threats and risks* must be systematically identified and analyzed. This activity involves conducting preventive risk assessments and threat modeling to understand the vulnerabilities present within the domain and prioritize their mitigation. Here, established frameworks, such as STRIDE, DREAD, SQUARE (Achour et al., 2015b), or Magerit (Mellado et al., 2010), provide robust guidelines. Other, less complex approaches may also be feasible, including UML modeling, activity models (Fernandez et al., 2015), or trade-off analyses (Sánchez et al., 2014).

Due to the increasing integration of *third-party services*, such as cloud services, they also play a significant role in security analysis. Many software systems rely on external services for functionality, introducing potential security vulnerabilities (May et al., 2023a). Evaluating these services for compliance, trustworthiness, and risks is essential to ensure that they do not become a weakness in the system's security. For instance, the integration of third-party components, such as data processors of Internet-of-Things devices, requires a careful assessment of their security features and compliance with domain-specific standards (Tomashchuk et al., 2021).

Security Design. Building on the findings of the security analysis, the security design process focuses on creating detailed specifications and strategies to address the identified security challenges and demands. This process ensures that security is embedded into the system from the outset and not added as an afterthought in the context of an optional quality attribute or non-functional requirement. Note that this process includes comprehensive testing (i.e., verification, validation) of all activities.

One of the primary activities in security design is the adoption of *security standards*, heavily building on the results of the security analysis concerning legal regulations. By leveraging established frameworks such as ISO/IEC 27001 (Mellado et al., 2010), the NIST cyber security framework (Wilson and Young, 2020), or the OWASP guidelines (Varela-Vaca et al., 2020), it can ensured that their security design aligns with industry best practices. These standards provide a foundation for defining security objectives (i.e., confidentiality, integrity, availability) and evaluating their implementation (Fægri and Hallsteinsen, 2006).

Security Requirements are another critical activity of this process. Derived from the domain analysis, these requirements specify what the system must achieve to protect itself against identified threats and risks. Examples include ensuring the goals of information security, i.e., confidentiality, integrity, and availability (Fægri and Hallsteinsen, 2006). These goals typically lead to associated requirements to be fulfilled, for example, through encryption (Fernandez et al., 2015), access controls (Fernandez et al., 2016), or redundancy mechanisms (Fægri and Hallsteinsen, 2006). Security requirements can be represented and prioritized in different ways, ranging from decision or feature models (Mellado et al., 2010) over goal-based models (Kim et al., 2008) and reusable patterns (Fernandez et al., 2016) to artifact annotations and con-

Table 1: Overview of the security-engineering process (italic) and activity fulfillment.

	Security analysis				*Security design*						*Security testing*				
Reference	Domain legal regulations	Domain requirements	Domain threats and risks	Third-party services	Security standards	Security requirements	Data protection	Business processes	*Security policy modeling & design*	*Security realization*	Compliance	Configurations & dependencies	Incident management	Source code	*Documentation*
Iqbal et al. (2024)	○	○	●	●	○	●	●	○	●	●	○	●	○	○	●
May et al. (2023a)	●	●	●	●	●	●	●	●	●	●	●	●	○	●	●
Varela-Vaca et al. (2023)	○	○	●	●	●	●	○	○	●	●	●	●	○	○	●
Tomashchuk et al. (2021)	●	●	●	●	○	●	●	○	●	○	●	●	○	○	●
Varela-Vaca et al. (2021)	●	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Williams et al. (2020)	○	○	●	●	○	●	●	○	●	●	○	○	●	○	●
Navas et al. (2020)	○	●	●	●	●	●	○	○	●	○	●	●	○	○	●
Young et al. (2020)	○	○	●	●	●	●	○	●	●	○	●	●	○	○	●
Adejokun and Siok (2020)	○	○	●	●	●	●	●	○	●	○	●	●	○	○	●
Varela-Vaca et al. (2020)	○	○	●	●	●	●	●	○	○	●	●	●	○	●	●
ter Beek et al. (2020)	○	○	●	●	○	●	●	○	●	●	○	●	○	○	●
Wilson and Young (2020)	●	●	○	●	●	●	●	●	●	●	●	●	●	●	●
Horcas et al. (2019)	○	○	●	○	○	●	●	○	○	●	○	●	○	○	●
Varela-Vaca et al. (2019)	●	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Horcas et al. (2018)	●	○	○	●	○	●	●	●	●	●	●	●	○	○	●
Ya’u et al. (2018)	●	○	○	○	○	○	●	○	○	●	●	●	○	○	●
Sprovieri et al. (2017)	○	●	●	●	○	●	●	●	●	●	○	●	○	○	●
Alam et al. (2017)	○	●	○	●	●	●	●	●	●	●	●	●	●	●	●
Fernandez et al. (2016)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Sion et al. (2016)	○	○	○	●	○	●	○	○	○	●	○	●	○	○	●
Varela-Vaca and Gasca (2015)	○	●	●	●	●	●	●	●	●	●	●	●	○	○	●
Myllärniemi et al. (2015)	●	●	●	●	○	○	●	○	●	●	●	●	○	○	●
Achour et al. (2015b)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Achour et al. (2015a)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Fernandez et al. (2015)	●	●	●	●	●	●	●	○	●	●	●	●	●	○	●
Sánchez et al. (2014)	○	●	●	●	○	●	●	○	○	●	○	●	○	○	●
Mellado et al. (2014)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Hammani et al. (2014)	●	○	●	○	○	●	○	○	○	●	●	●	○	○	●
Horcas et al. (2013)	○	●	●	●	○	●	●	○	○	●	○	●	○	○	●
Villela et al. (2012)	○	●	●	●	○	●	●	○	○	○	○	●	○	○	●
Tan et al. (2012)	○	○	●	●	○	○	●	○	○	○	○	●	○	○	●
Murwantara (2012)	○	○	○	●	○	●	○	○	●	●	○	●	○	○	●
Guana and Correal (2012)	●	○	●	●	○	●	●	○	●	○	●	●	○	○	●
Yu et al. (2012)	○	○	●	○	○	○	○	○	○	○	○	●	○	○	●
Zhang et al. (2011)	○	○	○	○	○	●	●	○	○	●	○	●	○	○	●
Ines et al. (2011)	○	●	●	●	○	●	●	○	●	●	●	●	●	●	●
Trujillo et al. (2011)	●	●	●	●	●	●	●	○	○	●	●	●	○	○	●
Rodríguez et al. (2011)	○	●	●	●	●	●	○	○	●	●	●	●	○	○	●
Mellado et al. (2010)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Mellado et al. (2009)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Mellado et al. (2008c)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Mellado et al. (2008d)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Mellado et al. (2008b)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Mellado et al. (2008a)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Etxeberria and Sagardui (2008)	○	●	●	●	○	●	○	○	○	●	○	●	○	○	●
Kim et al. (2008)	○	●	●	●	○	●	●	●	○	●	○	●	○	○	●
Hundt et al. (2007)	○	○	○	●	○	●	●	○	●	●	○	●	○	○	●
Mellado et al. (2007)	○	●	●	●	●	●	●	○	●	●	●	●	○	○	●
Fægri and Hallsteinsen (2006)	●	●	●	●	●	●	●	○	●	●	●	●	●	○	●
ISO/IEC 27000 (2022)	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

● Fulfilled ○ Not fulfilled

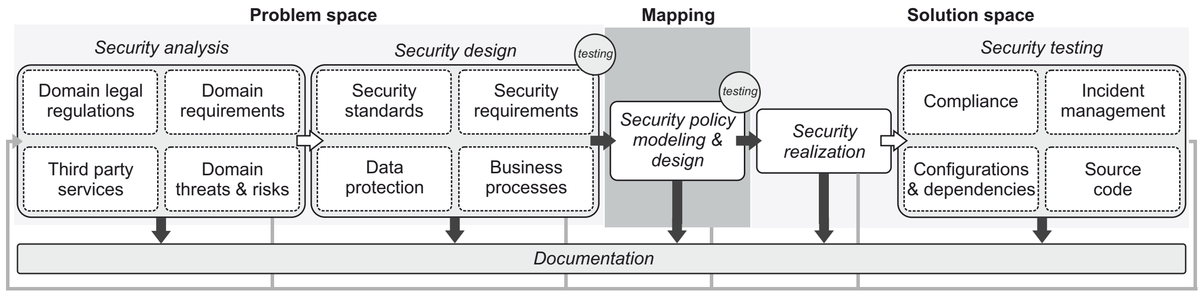

Figure 2: Overview of iterative security-engineering extension structured according to problem space, mapping, and solution space; including 18 characteristics, i.e., six processes (italic) and 12 activities.

straints (Villela et al., 2012).

Additionally, *Data protection* strategies are defined during the security design process, mainly referring to the specification of mechanisms related to data privacy, for example, using protocols including certain certificates (Mellado et al., 2008d) or enforcing encryption and authentication mechanisms (Varela-Vaca et al., 2019). Here, several relationships arise regarding the compliance with legal regulations, such as the GDPR (Tomashchuk et al., 2021).

Furthermore, security has to be considered regarding a system's *business processes*, such as controlling analyses (May et al., 2023a) or supply chain replenishment (Wilson and Young, 2020). This activity involves modeling workflows in Unified Modeling Language (UML) or as Business Process Model and Notation (BPMN), incorporating security steps without disrupting operational efficiency. Here, security measures should be designed so that they also mitigate risks. For example, Segregation of Duty (SoD), i.e., the separation of permissions in processes such as sales, ensures that a single role cannot simultaneously create orders, modify customer data, and approve invoices (Kobelsky, 2014). However, permissions and workflows must be designed in a way that maintains process efficiency. Paradigms such as role-based access control can help align security with efficiency.

4.2 Mapping

The mapping perspective serves as a bridge between the problem space and the solution space, involving abstracting the outcomes of the security analysis and design processes into actionable policies and models that guide the implementation of security measures.

Security Policy Modeling and Design. The fundamental process of the mapping is *security policy modeling and design*. It involves creating abstract models that define the system's security rules, goals, and constraints, including it's business processes. These are represented in different systematic ways. A more traditional way is related to security rules based on natural language (Varela-Vaca et al., 2019). Others refer to more logical representations, such as configuration rules (Guana and Correal, 2012), graph-related representations, such as feature models (Rodríguez et al., 2011) or attack trees to define constraints (ter Beek et al., 2020). As a further result of this phase, concrete reusable artifacts (Mellado et al., 2010) or architectural patterns (Fernandez et al., 2016) can be developed, serving as a reference for further core assets related to security. Note that comprehensive testing (i.e., verification, validation) of created policies is key to ensure their reliability.

4.3 Solution Space

The solution space focuses on implementing, verifying, and validation the security requirements and associated features defined in the previous processes. It includes two primary processes: security realization and security testing.

Security Realization. This process involves translating security policies and designs into practical implementations, including mapping security requirements to functional requirements related to the domain engineering (Etxeberria and Sagardui, 2008). In this way, a consolidation of existing features with the security requirements and their implementations is achieved. So, the process includes a variety of features to be implemented, for example, setting up firewalls, encryption protocols, access control lists, and other security mechanisms to align with the system's (security) design (Mellado et al., 2010). Additionally, business processes modeled in UML or as BPMN models should be transferred into XML format due to the shift of involved stakeholders. Furthermore, it might be useful to classify all implemented features according to their level of exploitation impact, making it easier to select a suitable risk acceptance according to stakeholder requirements (Etxeberria and Sagardui, 2008). One of the key tasks is also to ensure feature reusability while considering only valid and consistent configurations to avoid security misconfigurations (Myllärniemi et al., 2015).

Security Testing. Security testing verifies and validates the effectiveness and resilience of the implemented security design. This process involves testing legal compliance, configurations and dependencies, incident management capabilities, and source code.

The *compliance* activity includes verifying that the system adheres to legal, regulatory, and domain-specific security requirements (Mellado et al., 2007). Thus, compliance testing validates that all implemented security features are in line with the legal and regulatory obligations identified during the security analysis phase. For instance, testing might involve assessing whether data encryption, logging mechanisms, or access control policies (Fægri and Hallsteinsen, 2006) meet the standards outlined by legal regulations. In addition, it ensures that security practices evolve in response to updates in legal frameworks or domain standards (May et al., 2023a).

Testing *configurations and dependencies* is related to the verification of proper system configurations and dependencies to ensure that there are no security misconfigurations or overlooked vulnerabilities. This activity includes testing environments, tools, and third-party integrations (e.g., cross-configurations) to verify that they align with the security design (Varela-Vaca et al., 2021). For example, access control lists, encryption protocols, and dependency management systems are iteratively tested to identify weak configurations or exploitable points (Mellado et al., 2010). Additionally, dependencies such as third-party APIs or external libraries are evaluated to ensure that they do not introduce security risks (Horcas et al., 2018; Varela-Vaca et al., 2021).

Incident management testing focuses on the reusability and traceability of the system's underlying functions supporting the detection of and recovery from security incidents effectively. For instance, testing in the context includes verifying and validating system's logging and alerting functions as essential pillars in the overall vulnerability detection process, as well as the response mechanisms in place (Ines et al., 2011; Williams et al., 2020). Note, that we only focus on one part of the incident management process in this context. Typically, incident management relates more to a process after the actual development of a system, i.e. deployment and operation (ISO/IEC 27000, 2022). Nevertheless, we argue that testing reusability and traceability may increase its reliability sustainably.

Testing *source code* is a crucial activity to identify potential weaknesses, in particular, as developers' errors are a common problem. Typically, such programming errors are not deliberately reported, as responsible developers feel a certain amount of guilt in this context (Dietrich et al., 2018; May et al., 2025). However, this makes it even more important to implement testing strategies oriented towards features, products, and system families. For example, testing might involve code review processes to identify reused components and verify that their functions meet the organization's security reference guidelines (Varela-Vaca et al., 2020). Another aspect is related to clown-and-own and planting, in particular regarding practices of copying or reusing code or patterns from external sources, such as Stack Overflow, GitHub, or large-language models. Here, developers may inadvertently introduce vulnerabilities by copying insecure code snippets or patterns, including improperly implemented cryptographic functions, insecure default configurations, or undocumented dependencies (May et al., 2025).

4.4 Documentation

In parallel to each process and its activities, a comprehensive documentation has to be made, at best based on one centralized repositories for different purposes (Varela-Vaca et al., 2021; May et al., 2023a). Note that our examples may be case-dependent and do not include all possible repositories.

In the problem space, security analysis findings are stored, such as domain-specific requirements, compliance regulations, threat models, and risk assessments (Williams et al., 2020; Tomashchuk et al., 2021). This documentation serves as a reference for understanding the security landscape and forms the basis for security design, whose information is also stored, for example, including security requirements and architectures (Guana and Correal, 2012; Myllärniemi et al., 2015).

During the mapping phase, the documentation evolves to include security policies, models (e.g., variability models, BPMN models), and abstract designs (Horcas et al., 2013; May et al., 2023a). At best, the documentation in this phase consolidates all findings, models, and designs into a single reference that guides developers, testers, and other stakeholders, facilitating consistency and efficiency across PLE.

In the solution space, the documentation focuses on implementation and testing artifacts. They include detailed records of configurations, dependencies, test cases, test results, and incident management protocols (Wilson and Young, 2020; Varela-Vaca et al., 2021). For example, reusable test scripts, validated configurations, and incident response plans are stored to streamline implementation and support audits. This documentation also contains feedback from testing and monitoring, which drives continuous and iterative improvement (Mellado et al., 2008a; Varela-Vaca and Gasca, 2015).

5 DISCUSSION

In the following, we describe a selection of major implications based on our results as well as challenges that still remain, leading to further research directions.

Consideration of Business Processes. The limited attention to business processes within existing security frameworks and workflows highlights a gap. Modern systems increasingly rely on process modeling, such as BPMN, to define workflows involving tasks, interfaces, and roles (May et al., 2023a). However, the integration of security considerations at this stage is often overlooked (cf. Table 1). Secure BPMN modeling ensures that access controls, role permissions, and interface protections are embedded early in the development lifecycle. By transforming BPMN into machine-readable formats like XML, supported by frameworks such as OWASP XML Security, these processes become secure and executable. This approach not only enhances security but also maintains operational efficiency and compliance with industry standards such as ISO/IEC 27000 (2022).

Incident Management in Early Development Phases. Incident management, while critical, is typically addressed in operational contexts post-deployment (Tøndel et al., 2014). This narrow focus neglects the importance of reusable, traceable, and early incident management mechanisms during development. By testing functions related to logging, alerting, and recovery in pre-deployment, systems can improve their ability to detect and respond to incidents. For example, not only security-critical systems (e.g., ERP systems) but also systems with special requirements to safety (e.g., cyber-physical systems in manufacturing) can benefit from such tests. We believe that business process and their models may also benefit in this context. For instance, a log file can be generated after completing a process step (e.g., order creation), or dependent systems can be configured to create logs at these points to enable recovery and enhance security. Overall, we argue that validating incident response templates as part of pre-deployment testing enhances preparedness and ensures that security measures remain robust throughout the development lifecycle.

Comprehensive Source-Code Testing. We argue that, although source-code testing is a well-known research field (Varela-Vaca et al., 2019), it is often neglected in the analyzed literature. Furthermore, testing is typically only conducted in feature-based or product-based ways. However, family-based testing might also be a feasible strategy, involving cross-configurations and -dependencies between systems of a larger product portfolio (May et al., 2024a). Testing efforts should also address novel challenges related to planting, such as vulnerabilities introduced through reused code from external sources, i.e., clone-and-own (Krüger and Berger, 2020). Due to the increasing application of large-language models, we expect an even growing challenge in this context. In general, technologies related to artificial intelligence may be particularly prone to misbehavior, leading to significant risks related to their reliability and trust (May et al., 2024b).

Standard Adoption and Compliance. Not surprisingly, the adoption of domain legal regulations and standards (e.g., manufacturing, healthcare) remains inconsistent, reflecting the challenges in aligning domain-specific needs with established PLE workflows (Kenner et al., 2021). Integrating periodic reviews of compliance measures into PLE ensures alignment with evolving regulations and industry practices, supporting the successful transfer of solutions from theory into practice. However, standards are often resource-intensive to implement, deterring their practical application in smaller-scale or resource-constrained projects. We argue that lightweight adaptations tailored to specific domains, combined with automated compliance tools, could streamline their adoption.

Third-Party Dependencies and Services. The reliance on third-party services introduces significant vulnerabilities, such as different levels of authentication, non-isolation, or the use of legacy systems with inadequate security protection functions (Benaroch, 2021). However, only few of the analyzed studies provide comprehensive strategies for managing these dependencies. We argue that standardized evaluation criteria for third-party components in PLE, including compliance and risk assessments, are essential (NIST SP 800-30r1, 2012). Furthermore, more preventive approaches rather than reactive approaches are needed to better meet the security objectives of the CIA triad (Samonas and Coss, 2014).

5.1 Threats to Validity

There are several potential threats that might impair the internal (i.e., confidence that variables caused effects) and external validity (i.e., generalizability of findings to other settings). These are outlined in the following.

Internal Validity. Threats to internal validity include potential biases from subjective decisions in defining meta-characteristics and revising the framework extension iteratively. While the literature selection relied on three major literature databases, we may not considered all relevant research. To mitigate this, we employed snowballing to minimize the probability of overlooking relevant literature. Additionally, the terminology in the analyzed papers may have been misinterpreted. To address this, ambiguous terms were discussed and clarified among the authors. Varying levels of detail in the publications posed another threat, which we addressed by ensuring that the first and second authors discussed the extracted data from the selected works to minimize potential knowledge bias. In general, threats to internal validity were addressed by adopting a systematic methodology (Nickerson et al., 2013) that integrates two complementary construction strategies (cf. Figure 1) while actively involving experienced researchers and practitioners throughout the identification, synthesis, and discussion process.

External Validity. The external validity is influenced by its narrow focus on secure PLE frameworks and workflows, potentially overlooking broader security approaches. Although the inclusion criteria ensured relevance, restricting the analysis to 49 papers may have excluded additional valuable perspectives. However, by including literature published between 2005 and 2024, we argue that we considered a sufficient time frame, while ensuring timeliness of the analyzed research. Furthermore, the lack of a practical evaluation limits the generalizability to real-world contexts. To address this limitation, a comprehensive evaluation is already planned for future research by validating the extension in both PLE and security communities.

6 RELATED WORK

Security in PLE is increasingly recognized within the variability community. For instance, Krieter et al. (2018) explored securely handling product lines in cloud environments using Intel SGX, while Kenner et al. (2020) proposed representing attack vectors through feature models. Additionally, Varela-Vaca et al. (2020, 2021, 2023) leveraged feature models for automated verification and management of system vulnerabilities and security requirements.

Beyond solutions for specific security concerns, the niche field of security-engineering frameworks and workflows in PLE has emerged, forming the foundation of our empirical-to-conceptual construction strategy (Nickerson et al., 2013). Three key approaches stand out: Fægri and Hallsteinsen (2006) introduced a reference architecture for security-focused decision support in domain engineering; Mellado et al. (2007) developed the SREPPLine framework for systematic security-requirements engineering, further refined until 2010 (Mellado et al., 2010). Varela-Vaca et al. (2019) created the CyberSPL framework for verifying security requirements, features, and compliance. A closely related work is by May et al. (2023a), who proposed a PLE framework for secure ERP systems. While sharing our focus on integrating security engineering as a third pillar in PLE, their work is specific of ERP systems and business processes.

Although prior work has partially integrated security and PLE (cf. Table 1), our ISO/IEC 27000-compliant extension provides a more holistic, domain-independent perspective. By systematically detailing processes, activities, and their relationships, we aim to enhance understanding of security engineering in PLE, treating security as an equally important cross-cutting concern.

7 CONCLUSION

In this paper, we presented an extensions of the PLE framework (Pohl et al., 2005), called *security engineering*. Our approach builds on a systematic development process (Nickerson et al., 2013), including the analysis of 49 papers related to secure PLE. Additionally, we relied on the ISO/IEC 27000 (2022) series to guarantee that the overall extensions complies with established standards. In this way, we aimed to facilitate the transfer of theoretical knowledge into business practice. Overall, the extension is currently domain-independent, but may be transferable to other domains (e.g., manufacturing) by applying domain-specific standards and regulations in the problem space. Besides the presented six processes and 12 activities of the extension, we derived further implications and challenges related to business processes, incident management, source-code testing, security standards, and third-party services.

In this context, we argue that further research is needed, for example, to address gaps in successfully handling legal regulations and security standards in PLE as well as to protect business processes and their variability throughout PLE development. To this end, we already planned several future studies, including comprehensive evaluations based on case studies and multi-stage expert interviews within the PLE and security communities. For the security evaluation within the planned case study, we will combine different approaches: Specifically, the established STRIDE framework (Khan et al., 2017) will be used to analyze potential threats during the design phase (e.g., business process modeling), while the Cyber Kill Chain (Yadav and Rao, 2015) will assess the framework's effectiveness against potential attack phases (e.g., source-code testing or logging).

REFERENCES

Abal, I., Melo, J., Stănciulescu, Ş., Brabrand, C., Ribeiro, M., and Wąsowski, A. (2018). Variability bugs in highly configurable systems: A qualitative analysis. *Transactions on Software Engineering*, 26(3):1–34.

Achour, I., Labed, L., and Ghezala, H. B. (2015a). Formalization of secure service oriented product line. In *International Conference on Software Technologies (ICSOFT)*, pages 1–8. IEEE.

Achour, I., Labed, L., and Ghezala, H. B. (2015b). Proposition of secure service oriented product line. In *International Symposium on Computers in Education (SIIE)*, pages 52–59. IEEE.

Adejokun, A. and Siok, M. F. (2020). Effective systems security requirements in product line engineering. *INSIGHT*, 23(3):26–30.

Alam, M. M., Khan, A. I., and Zafar, A. (2017). A secure framework for software product line development. *International Journal of Computer Applications, Foundation of Computer Science*, 975:33–40.

Apel, S., Batory, D., Kästner, C., and Saake, G. (2013). *Feature-oriented software product lines*. Springer.

Batory, D. (2004). Feature-Oriented Programming and the AHEAD Tool Suite. In *International Conference on Software Engineering (ICSE)*, pages 702–703. IEEE.

Benaroch, M. (2021). Third-party induced cyber incidents—much ado about nothing? *Journal of Cybersecurity*, 7(1):tyab020.

Berg, K., Bishop, J., and Muthig, D. (2005). Tracing software product line variability: from problem to solution space. In *South African Institute of Computer Scientists and Information Technologists (SAICSIT)*, pages 182–191.

Dietrich, C., Krombholz, K., Borgolte, K., and Fiebig, T. (2018). Investigating system operators' perspective on security misconfigurations. In *Conference on Computer and Communications Security (CCS)*, pages 1272–1289. ACM.

Etxeberria, L. and Sagardui, G. (2008). Evaluation of quality attribute variability in software product families. In *nternational Conference on Engineering of Computer-based Systems (ECBS)*, pages 255–264. IEEE.

Fægri, T. E. and Hallsteinsen, S. (2006). A software product line reference architecture for security. In *Software Product Lines*, pages 275–326. Springer.

Fernandez, E. B., Yoshioka, N., and Washizaki, H. (2015). Patterns for security and privacy in cloud ecosystems. In *International Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE)*, pages 13–18. IEEE.

Fernandez, E. B., Yoshioka, N., Washizaki, H., and Syed, M. H. (2016). Modeling and security in cloud ecosystems. *Future Internet*, 8(2):13.

Gomes, A. T. A., Ziviani, A., Correa, B. S. P. M., Teixeira, I. M., and Moreira, V. M. (2012). Splice: A software product line for healthcare. In *International Health Informatics Symposium (IHI)*, pages 721–726. ACM.

Guana, V. and Correal, D. (2012). Improving software product line configuration: A quality attribute-driven approach. *Information and Software Technology*, 55(3):541–562.

Hammani, F. Z., Rhanoui, M., and El Asri, B. (2014). Towards a variable non-functional requirements integration for component-based product line a generic approach. In *World Conference on Complex Systems (WCCS)*, pages 146–151. IEEE.

Horcas, J. M., Pinto, M., and Fuentes, L. (2013). Variability and dependency modeling of quality attributes. In *Conference Series on Software Engineering and Advanced Applications (SEAA)*, pages 185–188. IEEE.

Horcas, J.-M., Pinto, M., and Fuentes, L. (2018). Variability models for generating efficient configurations of functional quality attributes. *Information and Software Technology*, 95:147–164.

Horcas, J.-M., Pinto, M., and Fuentes, L. (2019). Software product line engineering: A practical experience. In *Systems and Software Product Line Conference (SPLC)*, pages 164–176. ACM.

Hundt, C., Mehner, K., Pfeiffer, C., and Sokenou, D. (2007). Improving alignment of crosscutting features with code in product line engineering. *Journal of Object Technology*, 6(9):417–436.

Iglesias, A., Lu, H., Arellano, C., Yue, T., Ali, S., and Sagardui, G. (2017). Product line engineering of monitoring functionality in industrial cyber-physical systems: A domain analysis. In *Systems and Software Product Line Conference (SPLC)*, pages 195–204. ACM.

Ines, A., Khadouma, S., Labed, L., and Ghezala, H. B. (2011). Towards a secure service oriented product line. In *International Conference on Software Engineering Research and Practice (SERP)*, pages 1–7. Citeseer.

Iqbal, M., Hafeez, Y., Almashfi, N., Alsirhani, A., Alserhani, F., Ali, S., Humayun, M., and Jamal, M. (2024). Enhancing secure development in globally distributed software product lines: A machine learning-powered framework for cyber-resilient ecosystems. *Computers, Materials & Continua*, 79(3).

ISO/IEC 27000 (2022). Information technology – Security techniques – Information security management systems. Standard, ISO.

Kenner, A., Dassow, S., Lausberger, C., Krüger, J., and Leich, T. (2020). Using variability modeling to support security evaluations: Virtualizing the right attack scenarios. In *International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS)*, pages 1–9. ACM.

Kenner, A., May, R., Krüger, J., Saake, G., and Leich, T. (2021). Safety, security, and configurable software systems: A systematic mapping study. In *Systems and Software Product Line Conference (SPLC)*, page 148–159. ACM.

Khan, R., McLaughlin, K., Laverty, D., and Sezer, S. (2017). Stride-based threat modeling for cyber-physical systems. In *ISGT-Europe*, pages 1–6. IEEE.

Kim, J., Park, S., and Sugumaran, V. (2008). DRAMA: A framework for domain requirements analysis and modeling architectures in software product lines. *Journal of Systems and Software*, 81(1):37–55.

Kitchenham, B. A., Budgen, D., and Brereton, O. P. (2015). *Evidence-based software engineering and systematic reviews*. CRC Press.

Kobelsky, K. W. (2014). A conceptual model for segregation of duties: Integrating theory and practice for manual and it-supported processes. *International Journal of Accounting Information Systems*, 15(4):304–322.

Krieter, S., Krüger, J., Weichbrodt, N., Sartakov, V. A., Kapitza, R., and Leich, T. (2018). Towards secure dynamic product lines in the cloud. In *International Conference on Software Engineering (ICSE)*, pages 5–8. ACM.

Krüger, J. and Berger, T. (2020). An empirical analysis of the costs of clone-and platform-oriented software reuse. In *International Conference on the Foundations of Software Engineering (ESEC/FSE)*, pages 432–444. ACM.

May, R., Alex, A., Suresh, R., and Leich, T. (2024a). Product-line engineering for smart manufacturing: A systematic mapping study on security concepts. In *International Conference on Software Technologies (ICSOFT)*, pages 323–330. SciTePress.

May, R., Biermann, C., Kenner, A., Krüger, J., and Leich, T. (2023a). A product-line-engineering framework for secure enterprise-resource-planning systems. In *International Conference on ENTERprise Information Systems (CENTERIS)*, pages 1–8. Elsevier.

May, R., Biermann, C., Krüger, J., Saake, G., and Leich, T. (2022). A systematic mapping study of security concepts for configurable data storages. In *Systems and Software Product Line Conference (SPLC)*, pages 108–119. ACM.

May, R., Biermann, C., Krüger, J., and Leich, T. (2025). Asking security practitioners: Did you find the vulnerable (mis)configuration? In *International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS)*, pages 1–10. ACM.

May, R. et al. (2023b). A systematic mapping study on security in configurable safety-critical systems based on product-line concepts. In *International Conference on Software Technologies (ICSOFT)*, pages 217–224. SciTePress.

May, R., Krüger, J., and Leich, T. (2024b). Sok: How artificial-intelligence incidents can jeopardize safety and security. In *International Conference on Availability, Reliability, and Security (ARES)*, pages 1–12. ACM.

Mellado, D., Fernández-Medina, E., and Piattini, M. (2007). Sreppline: Towards a security requirements engineering process for software product lines. In *International Workshop on Security In Information Systems (WOSIS)*, pages 220–232. SciTePress.

Mellado, D., Fernández-Medina, E., and Piattini, M. (2008a). Security requirements engineering process for software product lines: A case study. In *International Conference on Software Engineering Advances (ICSEA)*, pages 1–6. IEEE.

Mellado, D., Fernández-Medina, E., and Piattini, M. (2008b). Security requirements in software product lines. In *International Conference on Security and Cryptography (SECRYPT)*, pages 442–449. SciTePress.

Mellado, D., Fernandez-Medina, E., and Piattini, M. (2008c). Security requirements variability for software product lines. In *International Conference on Availability, Reliability, and Security (ARES)*, pages 1413–1420. IEEE.

Mellado, D., Fernández-Medina, E., and Piattini, M. (2008d). Towards security requirements management for software product lines: A security domain requirements engineering process. *Computer Standards & Interfaces*, 30(6):361–371.

Mellado, D., Fernández-Medina, E., and Piattini, M. (2010). Security requirements engineering framework for software product lines. *Information and Software Technology*, 52(10):1094–1117.

Mellado, D., Mouratidis, H., and Fernández-Medina, E. (2014). Secure tropos framework for software product lines requirements engineering. *Computer Standards & Interfaces*, 36(4):711–722.

Mellado, D., Rodríguez, J., Fernández-Medina, E., and Piattini, M. (2009). Automated support for security requirements engineering in software product line domain engineering. In *International Conference on Availability, Reliability, and Security (ARES)*, pages 224–231. IEEE.

Mesa, O., Vieira, R., Viana, M., Durelli, V. H. S., Cirilo, E., Kalinowski, M., and Lucena, C. (2018). Understanding vulnerabilities in ppugin-based web systems: An exploratory study of wordpress. In *Systems and Software Product Line Conference (SPLC)*, pages 149–159. ACM.

Murwantara, I. M. (2012). Hybrid anp: Quality attributes decision modeling of a product line architecture design. In *International Conference on Uncertainty Reasoning and Knowledge Engineering (URKE)*, pages 30–34. IEEE.

Myllärniemi, V., Raatikainen, M., and Männistö, T. (2015). Representing and configuring security variability in software product lines. In *International Conference on the Quality of Software Architectures (QoSA)*, pages 1–10. ACM.

Navas, J., Voirin, J.-L., Paul, S., and Bonnet, S. (2020). Towards a model-based approach to systems and cybersecurity: Co-engineering in a product line context. *INSIGHT*, 23(3):39–43.

Nickerson, R. C., Varshney, U., and Muntermann, J. (2013). A method for taxonomy development and its application in information systems. *European Journal of Information Systems*, 22(3):336–359.

NIST SP 800-150 (2016). Guide to cyber threat information sharing. Standard, National Institute of Standards and Technology.

NIST SP 800-30r1 (2012). Guide for conducting risk assessments. Standard, National Institute of Standards and Technology.

Pohl, K. et al. (2005). *Software product line engineering: Foundations, principles, and techniques*. Springer.

Reinhartz-Berger, I., Cohen, S., Bettin, J., Clark, T., and Sturm, A. (2013). Domain engineering. *Product Lines, Languages and Conceptual Models*.

Rodríguez, J., Fernández-Medina, E., Piattini, M., and Mellado, D. (2011). A security requirements engineering tool for domain engineering in software product lines. In *Non-Functional Properties in Service Oriented Architecture: Requirements, Models and Methods*, pages 73–92. IGI Global.

Samonas, S. and Coss, D. (2014). The CIA strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security*, 10(3).

Sion, L., Van Landuyt, D., Yskout, K., and Joosen, W. (2016). Towards systematically addressing security variability in software product lines. In *Systems and Software Product Line Conference (SPLC)*, pages 342–343. ACM.

Sprovieri, D., Argyropoulos, N., Souveyet, C., Mazo, R., Mouratidis, H., and Fish, A. (2017). Security alignment analysis of software product lines. In *International Conference on Enterprise Systems (ES)*, pages 97–103. IEEE.

Sánchez, L. E., Diaz-Pace, J. A., Zunino, A., Moisan, S., and Rigault, J.-P. (2014). An approach for managing quality attributes at runtime using feature models. In *Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS)*, pages 11–20. IEEE.

Tan, L., Lin, Y., and Ye, H. (2012). Modeling quality attributes in software product line architecture. In *World Congress on Engineering and Technology (SCET)*, pages 1–5. IEEE.

ter Beek, M. H., Legay, A., Lafuente, A. L., and Vandin, A. (2020). Variability meets security: Quantitative security modeling and analysis of highly customizable attack scenarios. In *International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS)*, pages 1–9. ACM.

Thüm, T. et al. (2014). A classification and survey of analysis strategies for software product lines. *Computing Surveys*, 47(1):1–45.

Tomashchuk, O., Van Landuyt, D., and Joosen, W. (2021). The architectural divergence problem in security and privacy of ehealth iot product lines. In *Systems and Software Product Line Conference (SPLC)*, pages 114–119. ACM.

Tøndel, I. A., Line, M. B., and Jaatun, M. G. (2014). Information security incident management: Current practice as reported in the literature. *Computers & Security*, 45:42–57.

Trujillo, S., Alonso, I., Hamid, B., Gonzalez, D., Blanco, M., and Zhang, H. (2011). Towards variability support for security and dependability patterns: a case study. In *Systems and Software Product Line Conference (SPLC)*, pages 1–4. ACM.

van der Linden, F. J. et al. (2007). *Software product lines in action*. Springer.

Varela-Vaca, A. J. et al. (2023). Feature models to boost the vulnerability management process. *Journal of Systems and Software*, 195:1–22.

Varela-Vaca, Á. J., Gasca, R., Ceballos, R., Gómez-López, M. T., and Bernaldez Torres, P. (2019). CyberSPL: A framework for the verification of cybersecurity policy compliance of system configurations using software product lines. *Applied Sciences*, 9(24):1–28.

Varela-Vaca, A. J. and Gasca, R. M. (2015). Formalization of security patterns as a means to infer security controls in business processes. *Logic Journal of the IGPL*, 23(1):57–72.

Varela-Vaca, Á. J., Gasca, R. M., Carmona-Fombella, J. A., and T., G.-L. M. (2020). Amadeus: Towards the automated security testing. In *Systems and Software Product Line Conference (SPLC)*, pages 1–12. ACM.

Varela-Vaca, Á. J., Rosado, D. G., Sánchez, L. E., Gómez-López, M. T., Gasca, R. M., and Fernández-Medina, E. (2021). Carmen: A framework for the verification and diagnosis of the specification of security requirements in cyber-physical systems. *Computers in Industry*, 132:103524.

Villela, K., Arif, T., and Zanardini, D. (2012). Towards product configuration taking into account quality concerns. In *Systems and Software Product Line Conference (SPLC)*, pages 82–90. ACM.

Williams, P., Moss, P., Bataller, S., and Hassell, S. (2020). Engineering a cyber resilient product line. *INSIGHT*, 23(3):17–21.

Wilson, B. and Young, B. (2020). Cyber secure and resilient approaches for feature based variation management. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 1–6. IEEE.

Yadav, T. and Rao, A. M. (2015). Technical aspects of cyber kill chain. In *International Symposium on Security in Computing and Communication (SSCC)*, pages 438–452. Springer.

Ya'u, B. I., Nordin, A., and Salleh, N. (2018). Requirements patterns structure for specifying and reusing software product line requirements. In *International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, pages 185–190. IEEE.

Young, B., Darbin, R., and Clements, P. (2020). System security engineering and feature-based product line engineering: A productive marriage. *INSIGHT*, 23(3):13–16.

Yu, D., Geng, P., and Wu, W. (2012). Constructing traceability between features and requirements for software product line engineering. In *Asia-Pacific Software Engineering Conference (APSEC)*, volume 2, pages 27–34. IEEE.

Zhang, G., Ye, H., and Lin, Y. (2011). Modelling quality attributes in feature models in software product line engineering. In *International Conference on Software Technologies (ICSOFT)*, pages 249–254. SciTePress.