IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 22, 2021, accepted February 15, 2021, date of publication February 26, 2021, date of current version March 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3062388

Systematic Mapping of the Literature on Secure Software Development

HERNAN NINA[1,2], (Senior Member, IEEE), JOSÉ ANTONIO POW-SANG[1], (Senior Member, IEEE), AND MÓNICA VILLAVICENCIO[3]

[1]Maestría en Informática, Pontificia Universidad Católica del Perú, Lima 15088, Peru
[2]Carrera Profesional de Ingeniería de Sistemas, Universidad de Lima, Lima 15023, Peru
[3]Facultad de Ingeniería en Electricidad y Computación, Escuela Superior Politécnica del Litoral, 090902 Guayaquil, Ecuador

Corresponding author: Hernan Nina (hernan.nina@pucp.edu.pe)

ABSTRACT The accelerated growth in exploiting vulnerabilities due to errors or failures in the software development process is a latent concern in the Software Industry. In this sense, this study aims to provide an overview of the Secure Software Development trends to help identify topics that have been extensively studied and those that still need to be. Therefore, in this paper, a systematic mapping review with PICo search strategies was conducted. A total of 867 papers were identified, of which only 528 papers were selected for this review. The main findings correspond to the Software Requirements Security, where the Elicitation and Misuse Cases reported more frequently. In Software Design Security, recurring themes are security in component-based software development, threat model, and security patterns. In the Software Construction Security, the most frequent topics are static code analysis and vulnerability detection. Finally, in Software Testing Security, the most frequent topics are vulnerability scanning and penetration testing. In conclusion, there is a diversity of methodologies, models, and tools with specific objectives in each secure software development stage.

INDEX TERMS Software development, security, requirements, design, construction, testing, vulnerability, systematic mapping review.

I. INTRODUCTION

A study by the US Department of Homeland Security shows that more than 90% of cyber-attacks are not due to defects in cryptography, networks, or hardware, but due to vulnerabilities generated in software development [1]. Furthermore, the increase in malicious activities targeting software products and software security weaknesses has become a significant problem for the software development process. Likewise, it is necessary to address security issues from the beginning of the software development life cycle (and in each phase) instead of facing them by creating patches when the software is in the production phase [2]–[4]; therefore, a comprehensive model is needed to adapt security activities to the software development process [6]. The secure software development process comprises software requirements security, software design security, software construction security, and software testing security. This process aims to enrich security requirements, use threat models methodologies during software design, and apply best security practices for coding, code reviews, and tests [5]. To increase the security level of software products, this process should be continuously updated [5], [7]; hence, studies showing the trends in methodologies, notations, tools, and techniques are required.

Moreover, a systematic mapping study with an updated overview of trends in secure software development is necessary to identify the most relevant topics in the secure software development process and to complement existing studies which have a variety of objectives and context. For example, some systematic mapping reviews focus on: agile methodologies' security requirements; specific applications (e.g. Cyber-physical System), particular methods (e.g. problem frames or threat analysis), quality assessment, SWOT analysis, among others [8]–[14].

This paper is structured as follows: Section II describes the research methodology, section III presents the results of the systematic mapping study, and section IV poses the conclusions and future works.

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Chi Chen.

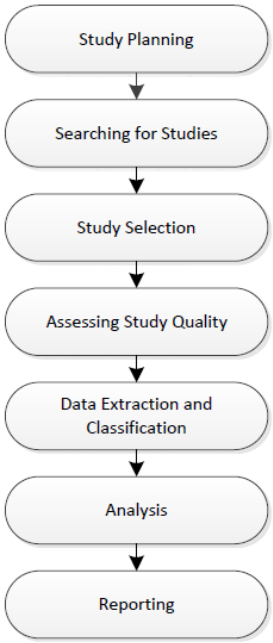

FIGURE 1. Process steps for systematic mapping studies.

II. RESEARCH METHODS

This systematic mapping follows the guidelines for performing systematic literature reviews in software engineering proposed by Kitchenham and Charters [15] and the guidelines for systematic mapping studies in security engineering recommended by Felderer and Carver [16]. Figure 1 illustrates the steps described to carry out systematic mapping studies.

Each process of the systematic mapping study in secure software development is presented below.

A. STUDY PLANNING

In this phase, the research questions must be formulated. To do so, the PICo technique [17] was applied, as follows:

- **Population:** Software development.
- **Interest:** Secure software.
- **Context:** The context corresponds to the sector of the Software development industry.

As a result, five research questions were stated for this mapping study (see Table 1).

B. SEARCHING FOR STUDIES

For the present study, the indexed bibliographies databases were used: Scopus, Web of Science, IEEE Xplore, and ACM Digital Library. These databases are frequently used in systematic mapping studies in the Software Engineering discipline and have automated search tools that allow an adequate level of coverage for the subject studied. Table 2 illustrates the selected sources for this mapping study.

The search for scientific articles in the above mentioned databases was carried out according to the following procedure:

TABLE 1. Research questions.

ID	Research questions
RQ1	What are the main trends in secure software development?
RQ2	What are the most prominent activities, and what methods are prevalent in software requirements security?
RQ3	What are the trends in software design security?
RQ4	What are the most frequent topics in software construction security?
RQ5	What techniques are prevalent at software testing security?

TABLE 2. Selected sources for the Systemic mapping study.

Source	Number of Studies	Sources indexed
Scopus	>70M	Cambridge University Press, Elsevier, Springer, Wiley-Blackwell, Nature Publishing Group y IEEE.
Web of Science	>33M	Web of Science, KCI-Korean, Russian Science, SciELO
IEEE	>5M	IEEE
ACM	>2M	ACM

- *Keywords:* Two keywords were obtained by decomposing the research questions: ''software development'' and ''secure software''.
- *Synonyms.* Synonyms were created for ''software development'' which correspond to the following terms: ''software engineering'', ''software process'', ''software construction'', and ''software project''. The following terms were used for the keyword ''secure software'': ''software security'', ''security requirement'', ''security attribute'', ''software vulnerability''.
- *Search strings.* Search strings were generated using logical operators: OR for synonyms and AND for combine keywords. Table 3 shows the string used in each database considering studies published in English between 2014 and 2019. The total number of publications found until June 10th, 2019 was 867.
- *Filtering articles.* Articles with a blank abstract or a language other than English were deleted.

C. STUDY SELECTION

For the final selection of the articles, the inclusion and exclusion criteria shown in Table 4 were used.

TABLE 3. Search strings by database.

Database	Search string
Scopus	TITLE-ABS (("software development" OR "software engineering" OR "software process" OR "software construction" OR "software project") AND ("software security" OR "security requirement" OR "security attribute" OR "software vulnerability" OR "secure software"))
Web of Science	TS= (("software development" OR "software engineering" OR "software process" OR "software construction" OR "software project") AND ("software security" OR "security requirement" OR "security attribute" OR "software vulnerability" OR "secure software"))
IEEE	(("software development" OR "software engineering" OR "software process" OR "software construction" OR "software project") AND ("software security" OR "security requirement" OR "security attribute" OR "software vulnerability" OR "secure software"))
ACM Digital Library	"query": { Title:(("software development" OR "software engineering" OR "software process" OR "software construction" OR "software project") AND ("software security" OR "security requirement" OR "security attribute" OR "software vulnerability" OR "secure software")) AND Abstract:(("software development" OR "software engineering" OR "software process" OR "software construction" OR "software project") AND ("software security" OR "security requirement" OR "security attribute" OR "software vulnerability" OR "secure software")) }

TABLE 4. Inclusion and exclusion criteria.

Type		Criterion
Inclusion	1.	Papers report at least one issue related to the requirements security, design security, secure code, and testing security applied in the context of a software development process.
	2.	Papers are in the field of software engineering.
	3.	Papers were peer-reviewed in conferences or journals.
	4.	Papers were published between 2014 and 2019
Exclusion	1.	Papers are not conducted in a secure software development context.
	2.	Papers are related to security teaching and experimentation with students.
	3.	Publications are not peer-reviewed or correspond to the abstract of a full book, an editorial, or a letter.
	4.	Papers are related to the protection of people and the privacy of their data.

TABLE 5. Summary of results.

Database	Search result	Duplicate articles	Relevant articles
Scopus	370	-	283
IEEEXplore	354	91	176
Web of Science	115	34	54
ACM DL	28	5	15
Total	867	130	528

After applying the inclusion and exclusion criteria and filtering duplicates, 528 articles were selected. Table 5 shows the summary of the search process.

For selecting the articles, the title and abstract were read. However, full-text reading was necessary in 49 studies.

D. ASSESSING STUDY QUALITY

Considering that a large number of studies were retrieved; one reviewer, the first author, filtered the papers applying the inclusion and exclusion criteria [18]. Next, actions were taken by the second author to evaluate the final set of articles to reduce the validity threat. The second author checked the extraction to reduce the bias. The quality assessment of selected studies was performed based on their credibility, integrity, and relevance to answer the research questions.

E. DATA EXTRACTION

The extraction of the relevant data from the selected studies was performed using a classification scheme (see Figure 2)

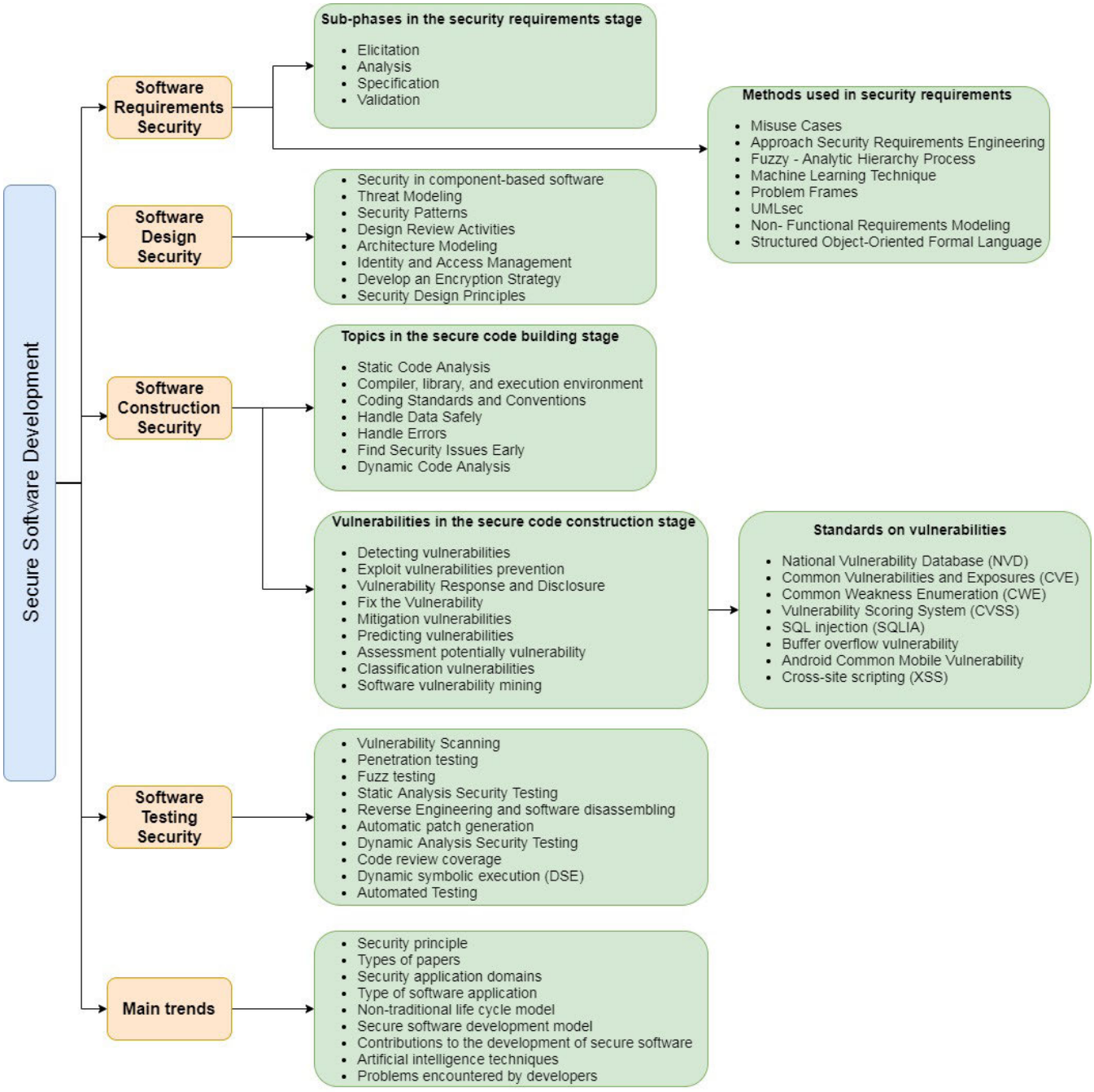

FIGURE 2. Classification scheme for data extraction.

based on a set of recommended rules and practices in secure software development obtained from: the Software Engineering Body of Knowledge (SWEBOK), ISO/IEC/IEEE 12207:2017 Standard, Software Security Assurance State of the Art Report (SOAR), and Software Assurance Forum for Excellence in Code (SAFECode).

III. RESULTS AND ANALYSIS

This section presents the results for each research question and consequently provides an overview of the secure software development process trends. A way to determine the relevance of security related aspects encountered in the context of software development can be through their frequency (i.e. the number of times that an aspect is reported in the analyzed documents).

A. RQ1: WHAT ARE THE MAIN TRENDS IN SECURE SOFTWARE DEVELOPMENT?

The main trends in the development of Secure Software according to systematic mapping are classified on the

TABLE 6. Number of articles by main trends.

Nº	Types of Papers	Frequency	Percentage
1	Phases of secure software development	267	42%
2	Security principle	42	8%
3	Types of papers	528	100%
4	Security application domains	64	12%
5	Type of software application	209	40%
6	Non-traditional life cycle model	46	9%
7	Secure software development model	44	8%
8	Contributions to the development of secure software	107	20%
9	Artificial intelligence techniques	52	10%
10	Problems encountered by developers	219	41%

following categories: 1) phases of secure software development, 2) security principles, 3) type of research studies, 4) application domains, 5) types of software, 6) types of non-traditional life-cycle models, 7) secure software development models, 8) contribution of research studies, 9) artificial intelligence techniques, and 10) security aspects according to developers. Table 6 shows the number of articles by main trends.

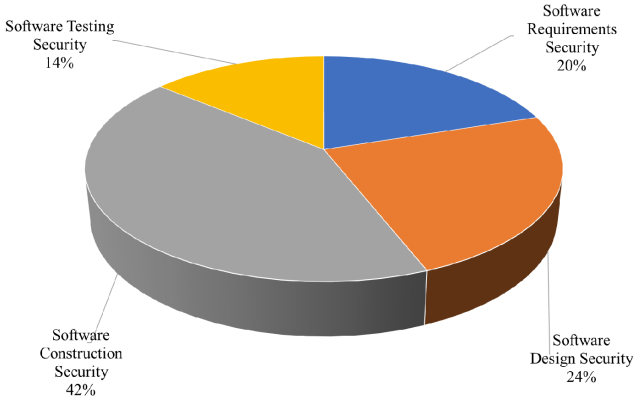

FIGURE 3. Classification by phases of secure software development.

First, the publications were classified based on the phases of secure software development. Likewise, according to the Software Engineering Body of Knowledge (SWEBOK) and CSSLP Certification, the phases of the construction of Secure Software correspond to the requirements, design, coding, and testing phase of secure software [19], [20]. Figure 3 illustrates that 267 (42%) of 528 studies focus on the coding phase of secure software, being this the first topic of interest, followed by the design, requirements, and testing phases. These results are similar to a previous systematic mapping study on security in the software development life cycle carried out with publications between 1980 and 2015, where coding, design, and requirements reached 42%, 29%, and 19%, respectively [14].

Second, the security principles were analyzed using the ISO/IEC/IEEE 12207-2017 standard and CSSLP Certification Program definitions – see Figure 4. The most frequently reported security principles are confidentiality, integrity, and availability. They were identified in 42 studies (8% of 528 articles) by reviewing the entire content of some articles.

Third, the articles were classified based on the type of types of papers recommended in [16] and [21], (see Table 7). As it can be observed, the evaluation research type is reported more frequently (53% of all reviewed papers). This kind of research implements a technique in practice or studies a problem in practice [16]. Only 23 papers reported systematic reviews and mappings.

Table 8 presents a closer view of the evaluation and validation research types, showing that a higher proportion corresponds to the case study research type followed by experiments.

Fourth, the articles were also sorted out by the security application domains, resulting in 64 studies; the predominant domain was government – see Table 9. Some of the articles propose government frameworks for secure software development (e.g. Information technology services to the Brazilian Government, Malaysian public sector for in-house web application development, Chinese IT security standards, etc).

Fifth, the type of software application was obtained from 209 articles that contained that information. Results are shown in Figure 5 where Web applications (33%) are the most referred since they are prey to constant attacks that are difficult to control. Also, large enterprise software presents a latent security concern because managing bulky assets, components, and a considerable amount of code is complex. Likewise, cyber-physical systems must be completely secure, since these are sensitive applications that integrate physical systems with software (e.g. systems that manage the operation of electrical networks). Similarly, emerging applications such as IoT applications are gradually becoming an attractive attack vector due to their limited security.

Sixth, the studies were classified according to the type of software life cycle model. To do so, it was necessary to review the entire content of some articles. Figure 6 shows that the most referred were the agile methodologies being

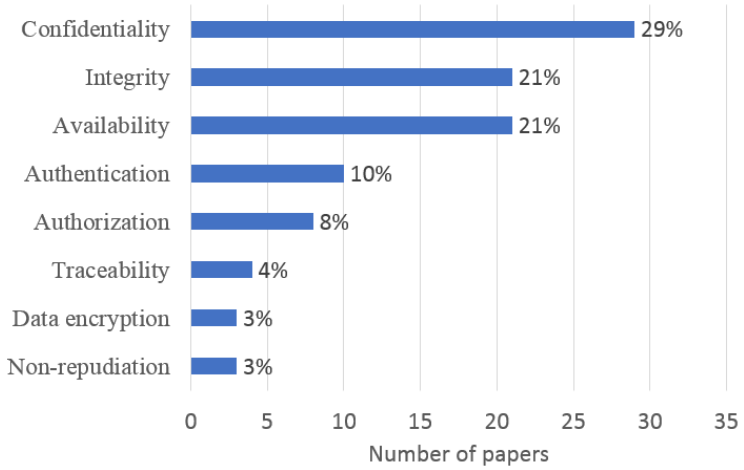

FIGURE 4. Frequency of papers by security principle.

TABLE 7. The trend in the types of studies related to secure software development.

Nº	Types of Papers	Frequency	Percentage
1	Evaluation Research	279	53%
2	Validation Research	127	24%
3	Proposed Solution	86	16%
4	Systematic review	12	2%
5	Systematic Mapping	11	2%
6	Opinion Research	7	1%
7	State of the art	6	1%
	Total	528	100%

TABLE 8. The evaluation research and validation research types reported.

Types of research	Evaluation Papers		Validation Papers	
Case study	136	49%	59	46%
Experiment	65	23%	58	46%
Survey	50	18%	5	4%
Discussion/argument	28	10%	5	4%
Total	279	100%	127	100%

TABLE 9. Security application domain.

Nº	Application Domain	Frequency	Percentage
1	Government	20	31%
2	Commercial industry	16	25%
3	Transport	13	20%
4	Health	10	16%
5	Banking and finances	5	8%
	Total	64	100%

the Scrum framework and the dynamic software development those that stand out. It was also found that agile methodologies are not the preferred option for developing large-scale projects, security-critical software projects, and projects where requirements are known in advance. Waterfall methodologies are preferred in those types of projects [22].

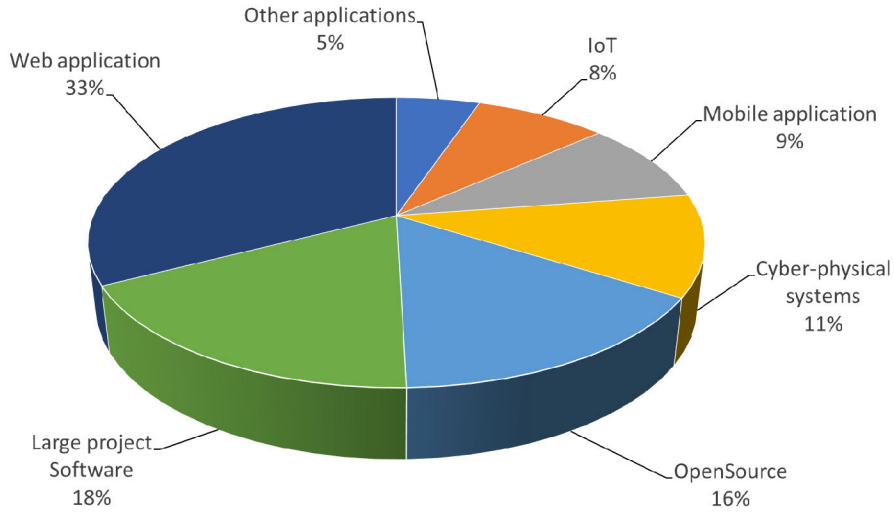

FIGURE 5. Classification of papers by type of application.

TABLE 10. The secure software development model.

Nº	Secure Software Development Model	Frequency	Percentage
1	OWASP - Comprehensive Lightweight Application Security Process (CLASP)	18	41%
2	MS Microsoft - Microsoft Security Development Lifecycle (SDL)	12	27%
3	NIST	5	11%
4	Gary MG - McGraw's Touchpoints	5	11%
5	ISO/IEC (such as 13335, 13849, 21434, 26262, 27002, or 27034)	4	9%
	Total	44	100%

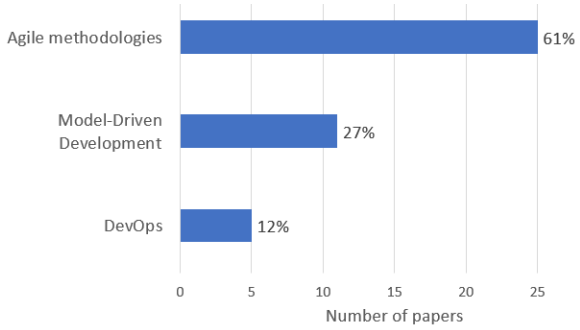

FIGURE 6. Classification of papers by non-traditional life cycle model.

Seventh, we looked for secure software development models among the articles. Table 10 illustrates the results obtained from reading 44 studies, where it can be observed that the Comprehensive Lightweight Application Security Process (OWASP - CLASP) and the Microsoft Security Development Lifecycle (SDL) are the most commonly referred. The models that do not appear in the studies despite their popularity are Security Apple Developer, Web Application Security Consortium (WASC), and SANS information security training. Apart from secure software development models, there are also organizations in the world that lead security issues such as Central Illinois Center of Excellence for Secure Software (CICESS), Illinois Central College (ICC), GDPR (General Data Protection Regulation), Malaysian Public Service Organization (MPS). Department of Defense (DoD), European Space Agency (ESA).

Eighth, the contributions that the researchers propose in their studies appear in Table 11. Models, methodologies, and tools are the most generally reported. A model is a graphic or mathematical description of a system with its respective properties; a methodology is a particular set of procedures; and a tool is the implementation of a process.

Ninth, we look at trends in the use of Artificial Intelligence techniques for secure software development. Figure 7 illustrates the result of 52 studies related to this issue. Machine learning techniques are undoubtedly trending in secure software development, among them the Bayesian network, Support Vector Machine (SVM), Long Short-Term Memory model (LSTM), K-Nearest Neighbor (KNN), Principal Component Analysis (PCA), Naive Bayes, and frequency -reverse document frequency TF-IDF.

Finally, Figure 8 presents the classification of papers according to the problems identified by developers regarding

TABLE 11. Contributions to the development of secure software.

Nº	Contributions in the Development of Secure Software	Frequency	Percentage
1	Models	36	34%
2	Methodologies	25	23%
3	Tools	19	18%
4	Framework	10	9%
5	Formal language	8	7%
6	Processes	9	8%
	Total	107	100%

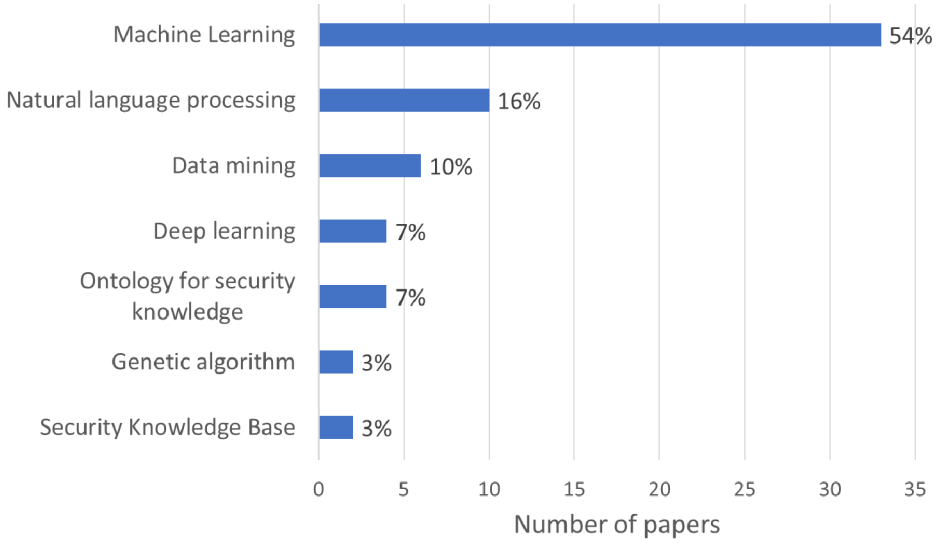

FIGURE 7. Classification of articles by artificial intelligence techniques.

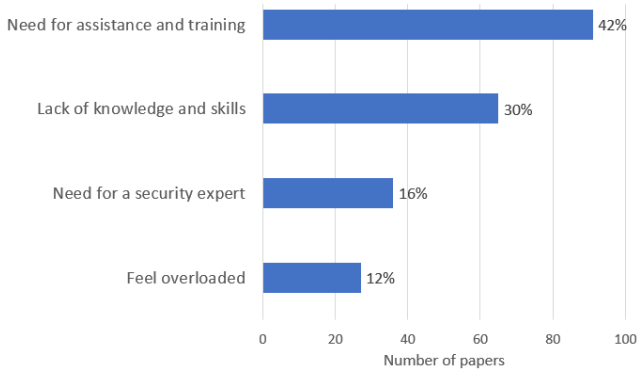

FIGURE 8. Problems encountered by developers.

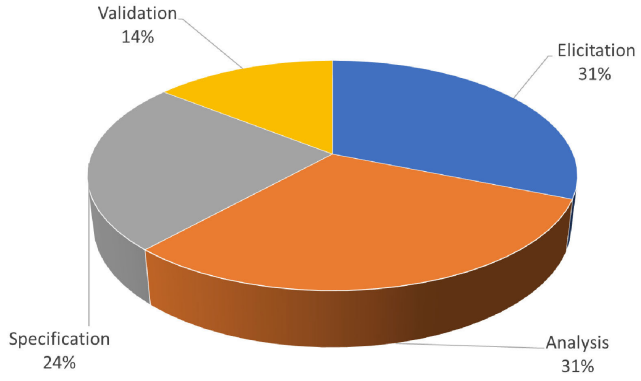

FIGURE 9. Sub-phases in the security requirements stage.

secure software development. As it can be seen, what developers require the most is assistance and training to develop secure software. This is understandable since the analyzed articles report that developers: lack of knowledge or skills to build these kind of applications, have the need for a security expert to guide them, and feel overloaded or have complex development activities to perform. Some proposals consider it essential to change the developer's mindset to make security a priority in software development. In this respect, security training helps, but developers find it challenging to apply it into their programming tasks [23]. It is important to consider that awareness of security issues is generated through several avenues, including company processes, standards, practices, and training, as well as the contextual factors that drive the focus on security [24].

TABLE 12. Sub-phases in the security requirements stage.

Nº	Sub-phases in the security requirements stage	Frequency	References (see Appendix A.)
1	Elicitation	48	[006],[029],[037],[038],[040],[046],[071],[112],[113],[144],[146],[147],[152],[188],[190],[205],[210],[212],[213],[214],[220],[222],[256],[274],[279],[294],[312],[313],[318],[339],[341],[361],[362],[366],[373],[375],[393],[394],[412],[418],[427],[445],[458],[478],[480],[499]
2	Analysis	47	[001],[017],[042],[044],[054],[090],[091],[109],[111],[112],[116],[121],[133],[134],[146],[156],[161],[164],[166],[170],[177],[180],[190],[205],[210],[213],[219],[240],[244],[245],[280],[286],[298],[312],[327],[333],[353],[363],[380],[410],[412],[430],[477],[480],[498],[502],[510]
3	Specification	37	[006],[037],[047],[112],[115],[127],[129],[131],[139],[151],[164],[195],[210],[223],[230],[239],[244],[256],[259],[271],[274],[286],[312],[325],[336],[359],[392],[394],[395],[445],[454],[456],[461],[476],[479],[480],[525]
4	Validation	22	[007],[031],[037],[060],[112],[113],[123],[147],[245],[305],[311],[312],[313],[327],[336],[341],[361],[362],[507],[523]
	Total	154	

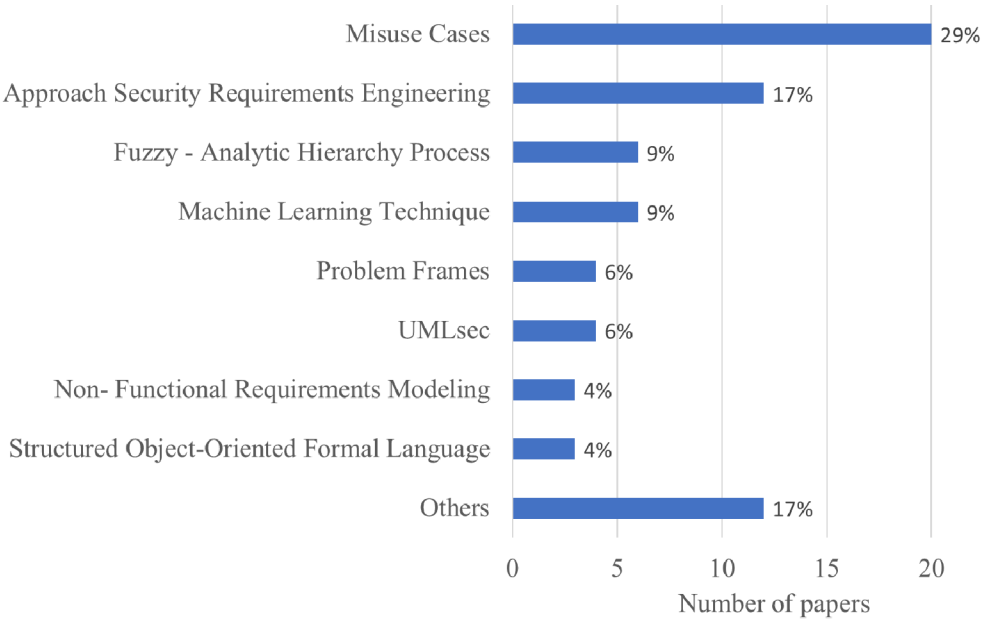

FIGURE 10. Methods used in security requirements.

B. RQ2: WHAT ARE THE MOST PROMINENT ACTIVITIES, AND WHAT METHODS ARE PREVALENT IN SOFTWARE REQUIREMENTS SECURITY?

Security requirements are divided into two categories. The first category includes software functions that implement security policies or that are required for security reasons. The second category is related to the probability that software security is threatened [19], [25]. Therefore, it is necessary to consider security aspects from the early phases of development. The first phase should identify the software security requirements and lay the foundation with high-level requirement models to continue developing secure software in the next phases [5].

In the secure software requirements phase, the sub-phases of elicitation, analysis, specification, and validation are carried out. Figure 9 and Table 7 shows the classification of 154 papers related to the requirement phase, being elicitation and analysis the sub-phases commonly referred. In the elicitation sub-phase, the sources of security requirements are identified based on the application's goals, domain, and stakeholders. For its part, the analysis sub-phase classifies and identifies limits, and detects and resolves conflicts between requirements. Finally, the specification sub-phase writes the requirements in a formal document to be evaluated in the requirements validation sub-phase.

Figure 10 and Table 13 presents the security requirements methods or techniques identified in 70 papers. These methods or techniques aim to identify precise requirements, without ambiguities, and balanced them with other software quality attributes such as usability and durability. The misuse

TABLE 13. Methods used in security requirements.

Nº	Methods used in security requirements	Frequency	References (see Appendix A.)
1	Misuse Cases	20	[023],[037],[038],[093],[111],[112],[129],[166],[170],[172],[220],[222],[223],[248], [294],[298],[359],[392],[498],[510]
2	Approach Security Requirements Engineering	12	[029],[059],[060],[191],[305],[361],[374],[380],[393],[430],[456],[458],[471]
3	Fuzzy - Analytic Hierarchy Process	6	[007],[114],[194],[230],[239],[240]
4	Machine Learning Technique	6	[006],[144],[256],[271],[479],[480]
5	Problem Frames	4	[113],[121],[499],[502]
6	UMLsec	4	[070],[206],[294],[374]
7	Non- Functional Requirements Modeling	3	[230],[333],[476]
8	Structured Object-Oriented Formal Language	3	[047],[115],[116]
9	Others	12	[029],[041],[205],[212],[280],[294],[361],[374],[395],[418],[427],[476]
	Total	70	

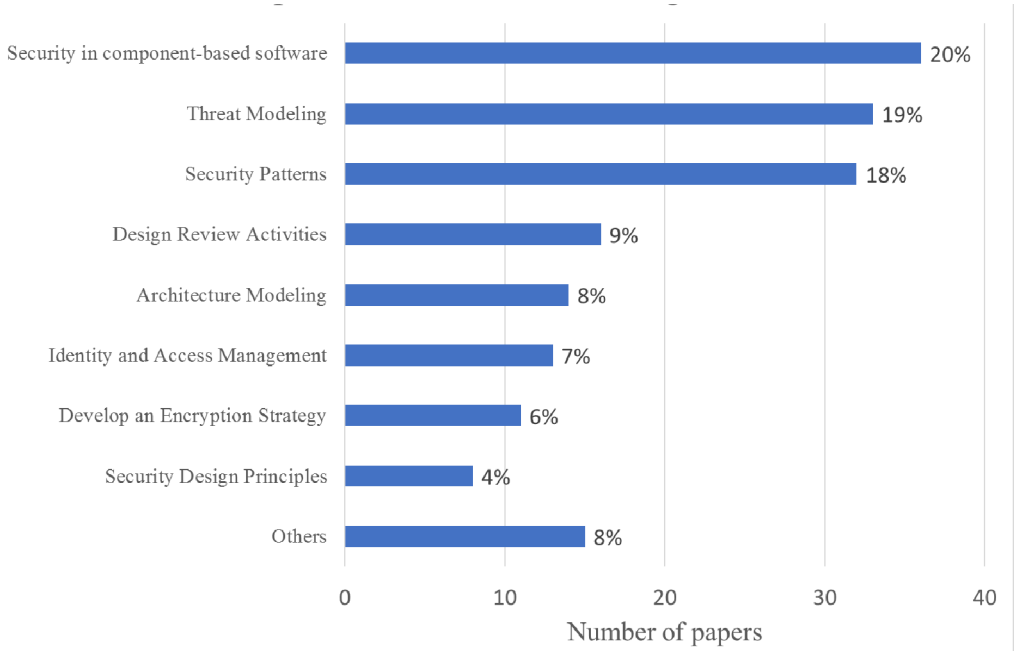

FIGURE 11. Security issues in the secure software design phase.

case is more frequently highlighted, which is used to elicit security requirements and identify potential threats in the use cases. Other methods or techniques were identified with the same objective and similar tasks, but with different names and approaches to tackle security. Likewise, many of the techniques derive from Security Requirements Engineering, but there is an interest in applying methods based on machine learning and agile methodologies. Also, for the prioritization of security requirements, the Fuzzy - Analytic Hierarchy Process method is reported. In addition, to represent security requirements, UML diagram extensions are used in techniques such as UMLsec. Finally, it is not a matter of choosing the best technique but looking how they can complement each other. For example, requirements for safety, and security, usability, and reliability.

C. RQ3: WHAT ARE THE TRENDS IN SOFTWARE DESIGN SECURITY?

According to the ISO/IEC/IEEE 12207-2017 standard definitions, software design consists of two activities: software architectural design and software detailed design. Software design security deals with the design of software modules to meet security objectives specified in the security requirements stage. Early attention to security issues, such as captured security requirements and design an architecture, can decrease the likelihood of weaknesses in design [25].

TABLE 14. Security issues in the secure software design phase.

Nº	Security issues in the secure software design phase	Frequency	References (see Appendix A.)
1	Security in component-based software	36	[008],[021],[072],[079],[080],[132],[141],[148],[156],[173],[195],[209],[210],[229],[231],[275],[276],[320],[321],[343],[357],[390],[402],[406],[414],[433],[444],[450],[472],[485],[486],[493],[503],[514],[516],[522]
2	Threat Modeling	33	[023],[029],[041],[042],[056],[113],[121],[127],[134],[139],[153],[155],[168],[170],[172],[294],[305],[307],[320],[334],[335],[340],[365],[388],[413],[422],[450],[453],[462],[463],[504],[513],[518]
3	Security Patterns	32	[017],[018],[027],[028],[044],[047],[068],[069],[080],[157],[158],[160],[248],[301],[349],[395],[414],[419],[429],[437],[438],[459],[460],[461],[462],[463],[464],[468],[474],[487],[488],[504]
4	Design Review Activities	16	[033],[114],[123],[153],[163],[183],[193],[205],[208],[225],[273],[311],[325],[427],[465],[477]
5	Architecture Modeling	14	[001],[002],[013],[030],[040],[055],[100],[111],[156],[225],[246],[298],[420],[430]
6	Identity and Access Management	13	[079],[100],[182],[275],[309],[347],[348],[396],[408],[409],[460],[508],[516]
7	Develop an Encryption Strategy	11	[002],[190],[236],[266],[273],[276],[309],[357],[387],[409],[467]
8	Security Design Principles	8	[096],[221],[277],[310],[349],[365],[369],[395]
9	Others	15	[028],[081],[128],[129],[181],[183],[205],[230],[287],[344],[353],[389],[409],[412],[429]
	Total	178	

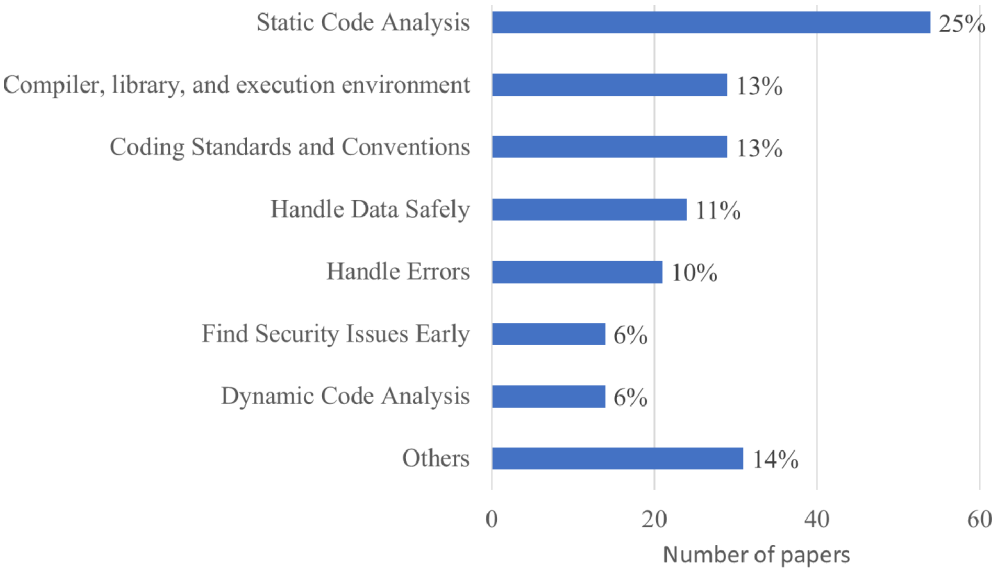

FIGURE 12. Topics in the secure code building stage.

In the present study, 149 papers (28% of the total sample) are related to secure software design topics. Figure 11 and Table 14 shows that security in a component-based software system, threat modeling, and security patterns are trends in design. Security in a component-based software system is an option to develop software using third-party software components since many times the security risks that can be inherited from such components are not taking into account, whether we use commercial off-the-shelf-products (COTS) or open-source software (OSS). Thread Modeling is an activity best executed in the software design phase and has the purpose of identifying the attacks that the software must withstand using the best defense strategy [26]. Security patterns correspond to structured solutions for recurring security problems and are reusable to design secure applications. Security Patterns allow software architects and designers to produce systems that meet their security requirements, are maintainable and extensible from the smallest to the most extensive systems. [27], [28].

Other topics are *design review activities* performed at the end of the detailed design stage and before starting the coding and testing phases [20], [25]. Architecture modeling includes the modeling of the architecture for comprehensive security analysis and the identification of possible attacks. Likewise, for architectural modeling, UML diagrams are used to represent entities, resources, privileges, safeguards, and

TABLE 15. Topics in the secure code building stage.

Nº	Topics in the secure code building stage	Frequency	References (see Appendix A.)
1	Static Code Analysis	54	[010],[016],[042],[049],[051],[063],[064],[104],[125],[134],[140],[153],[154],[168],[181],[196],[197],[198],[199],[215],[219],[236],[237],[262],[276],[281],[283],[284],[285],[295],[306],[307],[315],[329],[330],[331],[343],[352],[357],[377],[397],[399],[400],[407],[415],[416],[432],[443],[444],[448],[497],[501],[519],[523]
2	Compiler, library, and execution environment	29	[025],[081],[104],[115],[140],[145],[149],[180],[181],[196],[197],[213],[216],[258],[303],[343],[345],[357],[385],[398],[435],[451],[486],[490],[508],[515],[520]
3	Coding Standards and Conventions	29	[030],[042],[080],[131],[154],[186],[190],[228],[229],[248],[265],[328],[360],[405],[421],[423],[433],[436],[440],[461],[466],[481],[500],[501],[506],[514],[515]
4	Handle Data Safely	24	[062],[063],[085],[108],[133],[136],[145],[176],[177],[184],[231],[232],[252],[258],[330],[350],[364],[401],[405],[467],[482],[484],[526],[527]
5	Handle Errors	21	[093],[123],[191],[232],[236],[253],[257],[275],[290],[356],[371],[377],[419],[435],[441],[451],[452],[490],[503],[515],[517]
6	Find Security Issues Early	14	[016],[041],[077],[101],[105],[150],[216],[257],[284],[285],[310],[440],[443],[489]
7	Dynamic Code Analysis	14	[010],[045],[140],[174],[181],[236],[262],[295],[307],[330],[415],[441],[444],[446]
8	Others	31	[001],[002],[020],[022],[024],[025],[047],[065],[076],[107],[182],[207],[237],[238],[289],[303],[314],[320],[331],[333],[343],[352],[354],[357],[370],[399],[421],[426],[433],[448],[505],[519]
	Total	216	

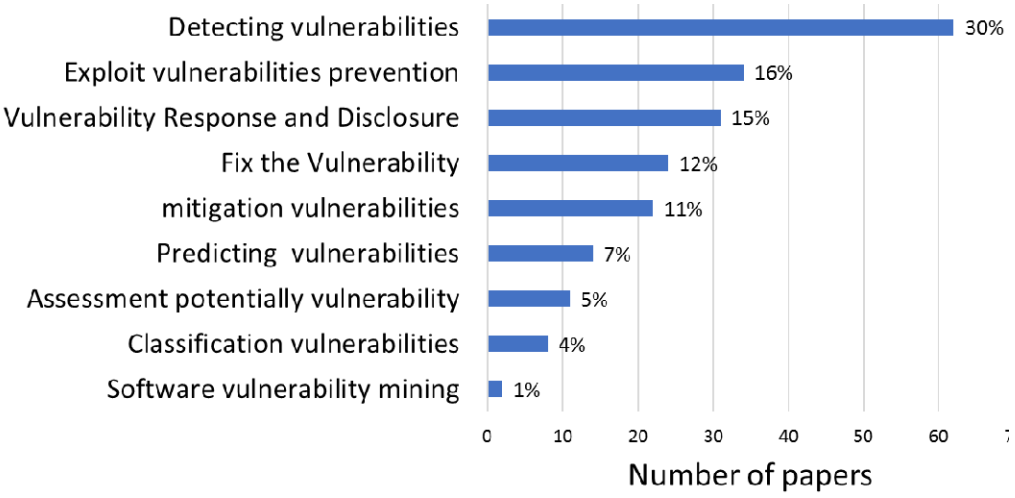

FIGURE 13. Vulnerabilities in the secure code construction stage.

policies [25]. Regarding identity and access management, the ISO 24760 standard defines them as a set of processes and policies involved in managing the life cycle of digital identity. Encryption Strategy is the encryption mechanism used to protect propagated data or inadvertent alteration, either of stored or transmitted data. This mechanism is considered comfortable, efficient, and cost-effective in the design process [26]. Finally, the security design principles establish general rules and guidelines for design.

D. RQ4: WHAT ARE THE MOST FREQUENT TOPICS IN SOFTWARE CONSTRUCTION SECURITY?

Software construction security refers to writing the code for specific situations to deal with security considerations. Encryption of security in software can be achieved by following the rules recommended in the Software Engineering Body of Knowledge (SWEBOK) [19]. These rules emphasize that the Software modules must be as small as possible, with validated and documented privileges, and with the ability to avoid sharing objects in memory with other programs. Figure 12 and Table 15 shows the classification of 160 studies related to secure code construction, where the most frequent topic corresponds to Static code analysis (25%). The classification is based on the approach made by SafeCode and the Information Assurance Technology Analysis Center (IATAC) [19], [20].

A vulnerability is a weakness expressed in errors or flaws in the software, which can be exploited by an attacker [29]. According to Andy Ozment, there are three leading causes of vulnerabilities in software: loss of motivation of

TABLE 16. Vulnerabilities in the secure code construction stage.

Nº	Vulnerabilities in the secure code construction stage	Frequency	References (see Appendix A.)
1	Detecting vulnerabilities	62	[010],[011],[032],[034],[036],[050],[051],[057],[064],[067],[071],[076],[077],[089],[101],[103],[109],[115],[125],[168],[169],[189],[200],[201],[204],[207],[210],[220],[228],[231],[237],[238],[253],[255],[260],[267],[270],[276],[281],[300],[307],[315],[329],[334],[361],[394],[421],[432],[433],[438],[457],[474],[483],[501],[505],[509],[512],[514],[517],[519],[520],[522]
2	Exploit vulnerabilities prevention	34	[022],[026],[034],[080],[083],[093],[107],[121],[130],[154],[175],[177],[178],[201],[204],[205],[236],[254],[261],[262],[289],[317],[320],[357],[396],[406],[433],[452],[457],[482],[493],[503],[512],[519]
3	Vulnerability Response and Disclosure	31	[005],[013],[014],[021],[073],[126],[199],[201],[202],[299],[304],[320],[329],[331],[332],[351],[383],[384],[386],[403],[406],[436],[439],[442],[469],[477],[492],[512]
4	Fix the Vulnerability	24	[010],[037],[042],[050],[051],[060],[082],[168],[192],[236],[249],[267],[284],[285],[300],[324],[340],[356],[362],[390],[396],[419],[443],[505]
5	Mitigation vulnerabilities	22	[026],[027],[039],[040],[125],[192],[208],[210],[220],[222],[238],[273],[306],[340],[341],[361],[405],[411],[432],[433],[447],[451]
6	Predicting vulnerabilities	14	[082],[089],[125],[200],[215],[247],[270],[278],[385],[407],[416],[428],[434],[437]
7	Assessment potentially vulnerability	11	[026],[027],[043],[054],[091],[247],[262],[331],[399],[506],[511]
	Classification vulnerabilities	8	[132],[178],[198],[281],[283],[390],[438],[497]
8	Software vulnerability mining	2	[254],[308]
	Total	208	

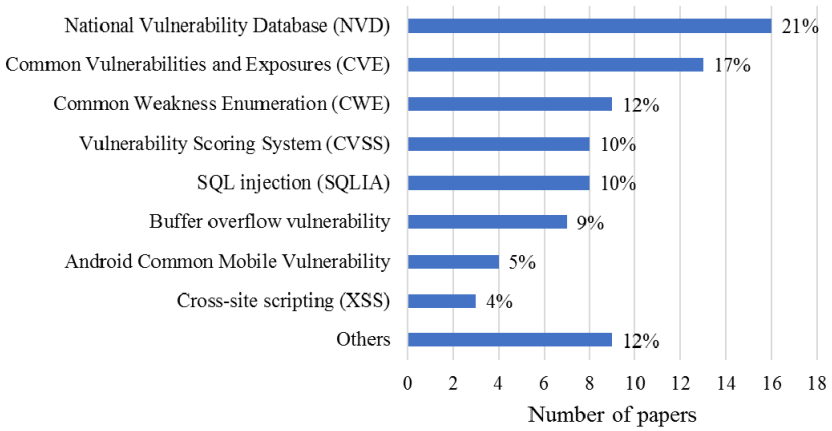

FIGURE 14. Standards on vulnerabilities.

TABLE 17. Standards on vulnerabilities.

Nº	Standards on vulnerabilities	Frequency	References (see Appendix A.)
1	National Vulnerability Database (NVD)	16	[012],[178],[199],[202],[203],[236],[247],[253],[304],[315],[324],[384],[390],[403],[442],[457]
2	Common Vulnerabilities and Exposures (CVE)	13	[043],[198],[199],[207],[253],[315],[317],[342],[357],[383],[442],[478],[505]
3	Common Weakness Enumeration (CWE)	9	[199],[200],[221],[236],[250],[315],[317],[396],[518]
4	Vulnerability Scoring System (CVSS)	8	[012],[090],[199],[202],[203],[308],[511],[512]
5	SQL injection (SQLIA)	8	[032],[078],[192],[236],[315],[334],[335],[400]
6	Buffer overflow vulnerability	7	[236],[261],[331],[357],[415],[435],[519]
7	Android Common Mobile Vulnerability	4	[283],[284],[285],[289]
	Cross-site scripting (XSS)	3	[236],[315],[385]
8	Others	9	[125],[126],[183],[258],[270],[457],[474],[513],[518]
	Total	77	

programmers due to constant changes in software requirements, lack of knowledge of programmers for developing complex software; and lack of use of adequate technology for the construction of secure software [25]. Figure 13 and Table 16 shows the classification of vulnerability issues in 159 studies. As it can be noticed, the two most referred

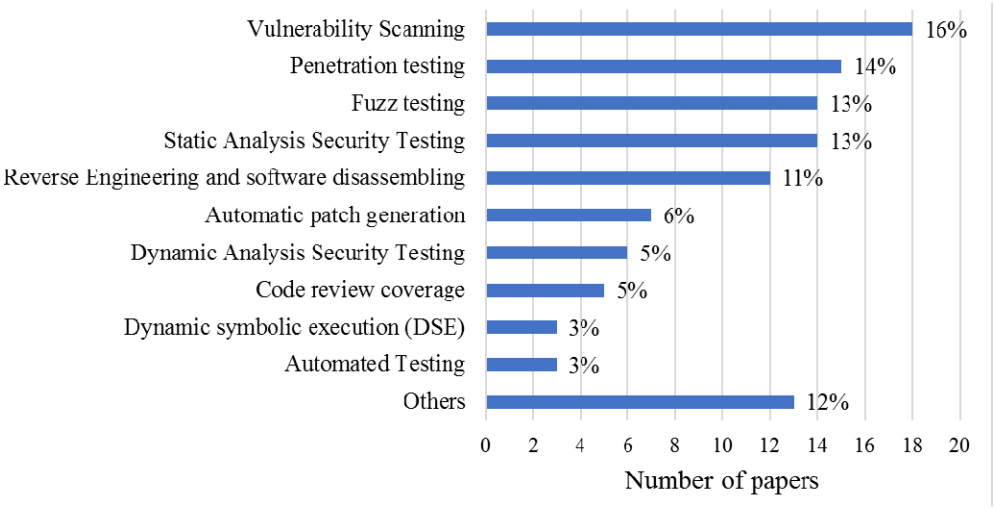

FIGURE 15. Topics in the secure software testing stage.

TABLE 18. Topics in the secure software testing stage.

Nº	Topics in the secure software testing stage	Frequency	References (see Appendix A.)
1	Vulnerability Scanning	18	[034],[042],[043],[061],[078],[082],[115],[134],[254],[267],[309],[361],[376],[431],[448],[469],[473],[505]
2	Penetration testing	15	[024],[042],[093],[122],[134],[168],[186],[189],[211],[222],[307],[358],[378],[403],[518]
3	Fuzz testing	14	[015],[042],[075],[076],[094],[119],[140],[179],[242],[307],[357],[402],[485],[520]
4	Static Analysis Security Testing	14	[024],[067],[074],[087],[103],[122],[219],[254],[342],[372],[378],[403],[449],[473]
5	Reverse Engineering and software disassembling	12	[025],[057],[101],[108],[120],[136],[142],[344],[345],[482],[527]
6	Automatic patch generation	7	[153],[207],[330],[352],[440],[482],[505]
7	Dynamic Analysis Security Testing	6	[042],[075],[122],[134],[140],[403]
8	Code review coverage	5	[035],[105],[426],[449],[455]
9	Dynamic symbolic execution (DSE)	3	[015],[035],[119]
10	Automated Testing	3	[153],[235],[501]
11	Others	13	[009],[015],[073],[084],[093],[123],[167],[223],[249],[326],[474],[514]
	Total	110	

issues are detecting vulnerabilities and exploit vulnerabilities prevention.

We also reviewed 57 articles to identify standards of vulnerabilities; the results appear in Figure 14 and Table 17. The National Vulnerability Database (NVD) and the Common Vulnerabilities and Exposures (CVE) are the ones that appeared most in the studies. Another noteworthy fact is that the vulnerability reports show that the vulnerabilities are related to common mistakes that are made during the programming phase [24].

E. RQ5: WHAT TECHNIQUES ARE PREVALENT AT SOFTWARE TESTING SECURITY?

Software security tests verify that the software protects data and complies with the implementation of the Software security requirements. There is a diversity of techniques and tools in the software security testing stage, among which the most notable are vulnerability scanning, penetration tests, risk assessment, security audit, ethical hacking, posture assessment, and safety regression [30]–[33]. The present study identified 83 papers related to software testing security; Figure 15 and Table 18 summarizes the findings.

IV. CONCLUSION AND FUTURE WORK

Unauthorized access to confidential data is frequent, and security threats grow exponentially. Hence the importance of developing secure software. However, in the literature it has been reported that secure software development approaches have weaknesses or are little or not accessible for their use in the software industry.

To deepen into these issues, this study provides an overview of trends in secure software development by reviewing 528 articles. We found out that the most frequently reported topics are software construction security or secure coding, evaluation research in security, case study papers, vulnerabilities in web applications, need for assistance and training in security for developers, and proposal of new security software models. The topics less frequently reported are

security principles, agile techniques and methods, popular secure software development models (OWASP or Microsoft SDL), and artificial intelligence techniques. Specifically, in the security requirements stage, the most frequently reported are elicitation and analysis of security requirements and the Misuse case technique. Recurring themes in secure software design are security in component-based software development, threat modeling, and security patterns. The secure code construction stage reported most frequently to static code analysis, vulnerability detection, and public vulnerability database (NVD). In the software security testing stage, vulnerability scanning and penetration testing are the most recurrent topics.

Finally, to know the characteristics of a secure software development method, technique, or tool, it is necessary to deepen the study by conducting systematic reviews.

APPENDIX

Papers Identified in the Systematic mapping of the literature on Secure Software Development

This information is available at:

http://inform.pucp.edu.pe/~jpowsang/ssd/mapping_study_appendix.htm

REFERENCES

[1] E. Bodden, "State of the systems security," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng., Companion (ICSE-Companion)*, May 2018, pp. 550–551.

[2] X. Meng, K. Qian, D. Lo, H. Shahriar, M. A. I. Talukder, and P. Bhattacharya, "Secure mobile IPC software development with vulnerability detectors in Android studio," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2018, pp. 829–830, doi: 10.1109/COMPSAC.2018.00141.

[3] R. A. Khan and S. U. Khan, "A preliminary structure of software security assurance model," in *Proc. 13th Int. Conf. Global Softw. Eng.*, May 2018, pp. 132–135.

[4] M. Saito, A. Hazeyama, N. Yoshioka, T. Kobashi, H. Washizaki, H. Kaiya, and T. Ohkubo, "A case-based management system for secure software development using software security knowledge," *Procedia Comput. Sci.*, vol. 60, pp. 1092–1100, Jan. 2015.

[5] M. Busch, N. Koch, and M. Wirsing, "Evaluation of engineering approaches in the secure software development life cycle," in *Engineering Secure Future Internet Services and Systems* (Lecture Notes in Computer Science), vol. 8431, M. Heisel, W. Joosen, J. Lopez, and F. Martinelli, Eds. Cham, Switzerland: Springer, 2014, doi: 10.1007/978-3-319-07452-8_10.

[6] M. Ramachandran, "Software security requirements engineering: State of the art," in *Proc. Int. Conf. Global Secur., Saf., Sustainability*, 2015, pp. 313–322.

[7] M. Felderer, B. Katt, P. Kalb, J. Jürjens, M. Ochoa, F. Paci, T. T. Tun, K. Yskout, R. Scandariato, F. Piessens, "Evolution of security engineering artifacts: A state of the art survey," *Int. J. Secure Softw. Eng. IJSSE*, vol. 5, no. 4, pp. 48–98, 2014.

[8] H. Villamizar, M. Kalinowski, M. Viana, and D. M. Fernandez, "A systematic mapping study on security in agile requirements engineering," in *Proc. 44th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2018, pp. 454–461.

[9] A. Souag, R. Mazo, C. Salinesi, and I. Comyn-Wattiau, "Reusable knowledge in security requirements engineering: A systematic mapping study," *Requirements Eng.*, vol. 21, no. 2, pp. 251–283, Jun. 2016.

[10] P. H. Nguyen, S. Ali, and T. Yue, "Model-based security engineering for cyber-physical systems: A systematic mapping study," *Inf. Softw. Technol.*, vol. 83, pp. 116–135, Mar. 2017.

[11] S. Rehman, V. Gruhn, S. Shafiq, and Y. I. Inayat, "A systematic mapping study on security requirements engineering frameworks for cyber-physical systems," in *Proc. Int. Conf. Secur., Privacy Anonymity Comput., Commun. Storage*, Dec. 2018, pp. 428–442.

[12] P. Silva, R. Noël, M. Gallego, S. Matalonga, and Y. H. Astudillo, "Software development initiatives to identify and mitigate security threats: A systematic mapping," in *Proc. CIbSE*, 2016, pp. 257–270.

[13] S. Wu, C. Zhang, and F. Wang, "Extracting software security concerns of problem frames based on a mapping study," in *Proc. 24th Asia–Pacific Softw. Eng. Conf. Workshops (APSECW)*, Dec. 2017, pp. 121–125.

[14] N. M. Mohammed, M. Niazi, M. Alshayeb, and S. Mahmood, "Exploring software security approaches in software development lifecycle: A systematic mapping study," *Comput. Standards Interface*, vol. 50, pp. 107–115, Feb. 2017.

[15] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ., Keele, U.K., Durham Univ., Durham, U.K., Tech. Rep., 2007.

[16] M. Felderer and J. C. Carver, "Guidelines for systematic mapping studies in security engineering," in *Empirical Research for Software Security*. Boca Raton, FL, USA: CRC Press, 2017, pp. 47–68.

[17] C. Lockwood, Z. Munn, and K. Porritt, "Qualitative research synthesis: Methodological guidance for systematic reviewers utilizing meta-aggregation," *Int. J. Evidence-Based Healthcare*, vol. 13, no. 3, pp. 179–187, 2015.

[18] M. Petticrew and H. Roberts, *Systematic Reviews in the Social Sciences: A Practical Guide*. Hoboken, NJ, USA: Wiley, 2008.

[19] I. C. Society, P. Bourque, and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*, 3rd ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 2014.

[20] W. A. Conklin and D. Shoemaker, *CSSLP Certification All-in-One Exam Guide*, 2nd ed. New York, NY, USA: McGraw-Hill, 2019.

[21] J. C. Carver, M. Burcham, S. A. Kocak, A. Bener, M. Felderer, M. Gander, J. King, J. Markkula, M. Oivo, C. Sauerwein, and L. Williams, "Establishing a baseline for measuring advancement in the science of security: An analysis of the 2015 IEEE security & privacy proceedings," in *Proc. Symp. Bootcamp Sci. Secur.*, Apr. 2016, pp. 38–51.

[22] L. Siddique and B. A. Hussein, "Practical insight about choice of methodology in large complex software projects in Norway," in *Proc. IEEE Int. Technol. Manage. Conf.*, Jun. 2014, pp. 1–4, doi: 10.1109/ITMC.2014.6918615.

[23] D. Oliveira, M. Rosenthal, N. Morin, K.-C. Yeh, J. Cappos, and Y. Zhuang, "It's the psychology stupid: How heuristics explain software vulnerabilities and how priming can illuminate developer's blind spots," in *Proc. 30th Annu. Comput. Secur. Appl. Conf. ACSAC*, 2014, pp. 296–305.

[24] T. Lopez, H. Sharp, T. Tun, A. Bandara, M. Levine, and B. Nuseibeh, "'Hopefully we are mostly secure': Views on secure code in professional practice," in *Proc. IEEE/ACM 12th Int. Workshop Cooperat. Hum. Aspects Softw. Eng. (CHASE)*, May 2019, pp. 61–68, doi: 10.1109/CHASE.2019.00023.

[25] T. Winograd, H. L. McKinley, L. Oh, M. Colon, T. McGibbon, E. Fedchak, and R. Vienneau, *Software Security Assurance: A State-of-the Art Report (SOAR)*. Herndon, VA, USA: Information Assurance Technology Analysis Center (IATAC) Data and Analysis Center for Software (DACS), 2007.

[26] S. Simpson, "SAFECode whitepaper: Fundamental practices for secure software development 2nd edition," in *Proc. ISSE Securing Electronic Business Processes*, Wiesbaden, Germany, 2014, pp. 1–32.

[27] E. Rodriguez, "Security design patterns," in *Proc. 19th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, 2003.

[28] A. V. Uzunov, E. B. Fernandez, and K. Falkner, "Securing distributed systems using patterns: A survey," *Comput. Secur.*, vol. 31, no. 5, pp. 681–703, Jul. 2012.

[29] S. Barnum and A. Sethi, "Attack patterns as a knowledge resource for building secure software," in *Proc. OMG Softw. Assurance Workshop, Cigital*, 2007.

[30] Y.-H. Tung, S.-C. Lo, J.-F. Shih, and H.-F. Lin, "An integrated security testing framework for secure software development life cycle," in *Proc. 18th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Oct. 2016, pp. 1–4.

[31] V. V. Ribeiro, D. S. Cruzes, and G. H. Travassos, "A perception of the practice of software security and performance verification," in *Proc. 25th Australas. Softw. Eng. Conf. (ASWEC)*, Nov. 2018, pp. 71–80.

[32] P. Nidagundi and M. Uhanova, "Software application security test strategy with lean canvas design," in *Proc. IVUS Int. Conf. Inf. Technol.*, Kaunas, Lithuania, Apr. 2018.

[33] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and Y. A. Pretschner, "Security testing: A survey," in *Advances in Computers*, vol. 101. Amsterdam, The Netherlands: Elsevier, 2016, pp. 1–51.

HERNAN NINA (Senior Member, IEEE) received the B.Sc. degree in informatics and system engineering and the Master of Administration degree from the Universidad Nacional de San Antonio Abad del Cusco (UNSAAC), Peru, the master's degree in informatica with mention in software engineering from the Pontificia Universidad Católica del Perú (PUCP), and the Ph.D. degree in system engineering from Universidad Nacional Federico Villareal (UNFV), Peru. He was qualified as a Researcher from CONCYTEC, Peru. He is currently a Full Professor with the Universidad de Lima, Peru. His research interests include software engineering, innovation in higher education, and human–computer interaction. He is also a member of ACM.

JOSÉ ANTONIO POW-SANG (Senior Member, IEEE) received the B.Sc. degree in informatics engineering and the Licentiate degree in education for development from the Pontificia Universidad Católica del Perú (PUCP), and the master's degree in software engineering and the Ph.D. degree in informatics engineering from the Universidad Politécnica de Madrid, Spain. He was the Executive Director of the Postgraduate School from 2013 to 2020 and the Director of the Master's Program in Informatics from 2011 to 2020. He is currently a Full Professor with PUCP. He has published several articles in the field of effort estimation for software development projects. His research interests include empirical software engineering, software metrics, software engineering education, and human–computer interaction. He is a member of ACM and the IEEE Computer Society. He has been elected President of the Peruvian Section of the IEEE Computer Society from 2015 to 2018.

MÓNICA VILLAVICENCIO received the Ph.D. degree in applied engineering from the École de technologie Súperieure, UQAM, Montreal, QC, Canada. She is currently a Full Professor with the Faculty of Electrical and Computer Engineering, Escuela Superior Politécnica del Litoral (ESPOL), Ecuador, where she teaches software engineering-related courses to undergraduate and graduate students. She is also the Director of the Doctorate Program of Applied Computer Science, ESPOL. Her research interests include agile software development, software measurement, software engineering education, and applied software engineering to the IoT and intelligent systems.