



## Security Aspect in Software Testing Perspective: A Systematic Literature Review

Halim Wildan Awalurahman<sup>1)</sup>, Ibrahim Hafizhan Witsqa<sup>2)</sup>,  
Indra Kharisma Raharjana<sup>3)\*</sup> , Ahmad Hoirul Basori<sup>4)</sup> 

<sup>1)2)3)</sup>Information Systems, Faculty of Science and Technology, Universitas Airlangga, Indonesia  
Jl. Dr. Ir. H. Soekarno, Mulyorejo, Surabaya

<sup>1)</sup>halim.wildan.awalurahman-2019@fst.unair.ac.id, <sup>2)</sup>ibrahim.hafizhan.witsqa-2019@fst.unair.ac.id, <sup>3)</sup>indra.kharisma@fst.unair.ac.id

<sup>4)</sup>Faculty of Computing and Information Technology in Rabigh, King Abdulaziz University, Rabigh, Saudi Arabia

<sup>4)</sup>abasori@kau.edu.sa

---

### Abstract

**Background:** Software testing and software security have become one of the most important parts of an application. Many studies have explored each of these topics but there is a gap wherein the relation of software security and software testing in general has not been explored.

**Objective:** This study aims to conduct a systematic literature review to capture the current state-of-the-art in software testing related to security.

**Methods:** The search strategy obtains relevant papers from IEEE Xplore and ScienceDirect. The results of the search are filtered by applying inclusion and exclusion criteria.

**Results:** The search results identified 50 papers. After applying the inclusion/exclusion criteria, we identified 15 primary studies that discuss software security and software testing. We found approaches, aspects, references, and domains that are used in software security and software testing.

**Conclusion:** We found certain approach, aspect, references, and domain are used more often in software security testing

**Keywords:** Software security, Software testing, Security testing approach, Security threats, Systematic literature review

**Article history:** Received 16 January 2023, First decision 24 March 2023, Accepted 10 April 2023, Available online 28 April 2023

---

### I. INTRODUCTION

The importance of security features has increased as software systems play more crucial roles in many applications [1]. Due to the sheer volume of daily intrusions, security has grown to be a particularly difficult problem for software development organizations [2]. One of the reasons why this consequence has arisen is that vulnerabilities in software development account for more than 90% of cyberattacks rather than flaws in encryption, networks, or hardware [3]. Relevant to the issue, security testing can pro-actively detect program vulnerabilities [4].

A few studies have discussed the correlation between software security and software testing. However, most of them only cover specific domain and do not explore the correlation of software security and software testing as a whole. The issues raised were software security risks and practice related to software development [5], requirements testing on safety requirements [6], importance of software testability and robustness [7], and security testing in web applications [8].

In [5], the current state of software security risks is examined. Software security risks were classified using a systematic literature analysis, and security measures were recommended to save money, time, and effort while still producing secure software applications. Other studies [6] provide requirements testing examples for safety needs. The current state of knowledge regarding the significance of software testability and robustness [7] and security testing in web applications [8] provided a brief overview of testing and security in each of their respective contexts, but neither one of them revealed any clear connections between the two.

In those studies, we found that testing and security have an important relationship. However, due to the specific issue that they raised, we cannot see the clear picture of this relationship as a whole. Therefore, we see this as an

---

\* Corresponding author

opportunity to conduct a systematic literature review on the state of the art of software testing related to security aspect. Especially regarding security aspects and the methods to perform software testing to ensure the software security.

## II. LITERATURE REVIEW

### A. Software Testing

Software testing dynamically checks and confirms that a system or program acts as intended when put through a limited number of test cases, often chosen from an unlimited execution domain [7]. Functional testing and non-functional testing are the two types of testing. Whereas non-functional testing evaluates how the system performs, such as performance, usability, or security, functional testing determines whether the product satisfies its specification based on software requirements [9]. If flaws are discovered, testers reduce the risk by making people aware of them and offering fixes before a product is released [10]. To inform stakeholders about the quality of the product or service being tested, an investigation known as software testing is carried out. [11].

### B. Software Security

The process of building and developing software that ensures the integrity, confidentiality, and availability of its code, data, and service is known as software security [12]. It is described in the IEEE standard as the capability to stop software from being seriously flawed [13]. The ability to safeguard information in software or systems so that people, other products, or systems can access the data and exercise their power over it is what ISO/IEC described as [14], while McGraw defined it as the ability of software running correctly under the circumstance of being attacked maliciously [13].

### C. Software Security Testing

Software security testing looks for potential security flaws in the software as well as whether it satisfies both functional and security criteria [15]. Its objectives are to identify security flaws and confirm that the features are secure [8]. It may also be described as a procedure to check whether an information system is safeguarding data and continuing to perform as intended [11].

### D. Related Secondary Studies

There are secondary studies that explored software security and software testing. These studies were conducted separately, raising specific issue for each one. Software security threats and practices connected to software development were discussed [5], as well as the significance of software testability and robustness [7], requirements testing on safety requirements [6], and security testing in web applications [8]. These works are summarized in Table 1.

Khan et al. [5] argued that security in software development organizations is just an afterthought and hasn't been adequately addressed. They did a thorough analysis of the literature to identify key research to learn about software security risks and practices to achieve better development method. As the result, they are able to prescribe different security activities to follow the software development life cycle's several phases in order to deliver secure software products with the least amount of work, money, and time possible.

Dos Santos et al. [6] investigated the main approaches to requirement testing especially related to the context of safety-critical systems (SCS). Critical systems are those used in avionics, health, nuclear, automotive, aircraft, and military applications that pose a risk to human life or the environment. The result of this study is the primary methods for requirement testing and how they're used in both academics and business.

Hassan et al. [7] addressed the issue of the relation of software testability with other quality attributes, namely software robustness. This study aims to show state-of-the-art in terms of crucial concerns relating to software testability and software resilience. The result shows that the testability concerns with the most research are observability and controllability, and the robustness issues with the most research are fault tolerance, handling exceptions, and handling external influences.

Aydos et al. [8] pointed out that there is an increase in the number of studies related to security of web applications. This event has brought new challenges for practitioners or new researchers to get an overview of this area. Therefore, this study aims to sum up the current state of web application security testing.

The studies mentioned above clearly have their own focus and domain. All of them touched some points in both software security and software testing. All of them implicitly show the relation between the two. However, there are none that explored the relation between software security and software testing in general, especially in listing the aspect and approach regarding to software security and software testing.

TABLE 1  
 RELATED STUDIES

Reference	Goal	Concerns in research questions
[5]	Learn about software security threats and procedures to improve the design of secure software development.	A. Security pitfalls that software vendor companies should avoid when creating safe software applications B. The guidelines that vendors should adhere to while creating secure software applications
[6]	Examine the various methods for testing requirements, paying special attention to the safety requirements in the context of Safety Critical Systems (SCS)	A. The primary methods for testing requirements that have been suggested in the literature B. How requirements testing techniques are used in safety-critical requirements C. The benefits and drawbacks of various methods for testing safety requirements D. To what extent are these methods used by business professionals? E. Indicators that the strategy demonstrates cooperation between requirement engineers and testers
[7]	Present state-of-the-art with respect to issues of importance concerning software testability and an important quality attribute: software robustness	A. The most recent developments in testability and software robustness
[8]	Summarize the state-of-the-art in web application security testing which could benefit practitioners to potentially utilize that information	A. Different types of contribution and the primary contribution of studies on web application security testing in terms of procedures, instruments, processes, and metrics B. Different studies on Web Application Security Testing have utilized several types of research methodologies C. Security testing equipment testing D. Testing tool licensing type for academic and industrial researchers E. The use of static and dynamic evaluation in security testing

### III. METHODS

In order to prepare the SLR, we followed the steps in [16]–[18]. The processes cover planning, carrying out, and reporting on reviews. We also took into account [19], which offered recommendations and illustrations for doing systematic literature reviews in software engineering.

#### A. Review Planning

During the planning of the review, we specify the research topics. Then, according to the research inquiries, we select the search term, inclusion standards, and criteria for exclusion.

##### 1) Goals and Research Queries

The gap that we found in early researches urges the need to conduct research on security aspects and the methods to perform software testing to ensure the software security. To address these demands, we created the following study questions:

RQ1: What are the approaches that can be considered in testing software security?

RQ2: What are the aspects that can be considered in testing software security?

RQ3: What are the references that are used in the aspects and approaches for testing software security?

RQ4: How is software security testing conducted in different domains?

We chose RQ1 to examine methods that were available and have been applied to testing software security across a wide range of applications and domains. We chose RQ2 to examine the aspects, much like RQ1 did. We anticipate that RQ1 will provide instructions on how to do software security testing, and RQ2 will provide the tested components or elements. The methods and components used to assess software security are still dispersed throughout numerous scopes, applications, and domains. As a result, we listed and mapped them using RQ1 and RQ2. To discover references that, if any, highlighted the techniques and aspects, we added RQ3. As the source from which the techniques and aspects were derived, RQ3 will contribute. In the end, we opted to use RQ4 to show how RQ1, RQ2, and RQ3 are applied in their respective fields. We also intend to use RQ4 to examine the usage patterns of various approaches and aspects across various domains.

##### 2) Search Query

After finding objectives and research questions, we proceeded to formulate the search query. In order to find relevant studies, we selected databases and identified keywords. We also created a search string to make the search

easier and which covers all related categories, namely software security and software testing. These informations can be seen in Table 2 and Table 3.

TABLE 2  
 IDENTIFIED KEYWORDS

Category	Keywords
Software Security	software security
Software Testing	software testing

We combined the keywords to create a search string: (“software security” AND “software testing”). We used the search string in the following sources and paid attention to the types of items and language.

TABLE 3  
 SEARCH SOURCES

Electronic Databases	IEEE Xplore Science Direct
Searched items	Research articles
Language	English

### 3) Inclusion and Exclusion Criteria

Following the retrieval of the search query, we listed the inclusion and exclusion criteria. With the help of the inclusion and exclusion criteria, the pertinent studies were picked.

The study (I1) must be a peer-reviewed publication, (I2) be written in English, (I3) be related to the provided search terms, and (I4) have been published between 2008 and 2022 in order to meet the inclusion criteria.

Short papers (E1), doctoral symposium papers (E2), proposals, lecture notes (E3), editorials (E4), comments (E5), tutorials (E6), review papers (E7), and articles summarized from conference keynotes (E8) are all excluded.

### B. Conducting the Review

After the objective, research questions, search query, and established inclusion and exclusion standards, we proceeded to the main part of the review. In this section, we use all the information that we have collected to search studies, select them, and filter them to find the relevant studies that will be used. The outcomes of the study search and selection procedure are shown in this section. The quality assessment findings are also presented here.

#### 1) Research Search and Selection

We used IEEE Xplore and ScienceDirect as the source or the databases of our search. Using the search query and criteria that we have decided, we ran the search simultaneously to make the search faster. We read the title and abstract of each paper shown in search result and then decided if they matched our criteria. If they did, we then added them to our library for further filtering. The study search process is depicted in Fig. 1.

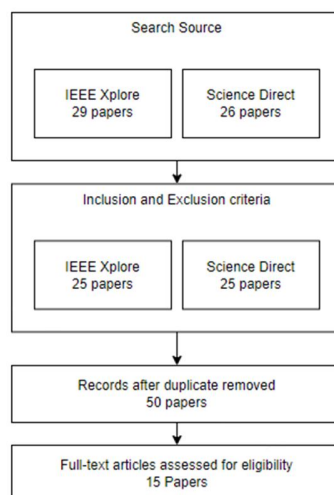


Fig. 1 Study Search Procedure

IEEE Xplore resulted in 29 papers while ScienceDirect resulted in 26 papers. We then filtered these papers using our inclusion and exclusion criteria. Some of these were excluded because they were included in our exclusion criteria.

The result is 25 papers from IEEE Xplore and 25 papers from ScienceDirect passed our inclusion and exclusion criteria.

After inclusion and exclusion criteria, we proceeded to check if duplicates existed. However, there were no duplicates as each of these libraries has its own database and are not related to each other. So, a total of 50 papers was found.

Our filtering did not stop at 50 papers. We continued to read the paper thoroughly in order to get more insight of the content. Some of the paper did not match our criteria as it only mentioned either software security or software testing which we will be discussing later in the discussion section. As a result, we selected only 15 papers that matched our criteria and could possibly answer our research questions.

TABLE 4  
 QUALITY ASSESSMENT FORM

Item	Assessment criteria	Score	Description
QA1	Was the goal of the study stated clearly?	-1	No
		0	Partially
		1	Yes
QA2	Is the proposed approach described in detail in the research?	-1	No
		0	Partially
		1	Yes
QA3	Is the suggested strategy proven to work?	-1	No
		0	Partially
		1	Yes
QA4	Is there a position or opinion expressed in the research?	-1	Yes
		0	Partially
		1	No
QA5	Do other scholarly publications reference the study?	-1	No
		0	Partially
		1	Yes

### 2) Quality Assessment

We carried out quality assessment to look at the primary studies' methodology. The evaluation of quality employed by [16] was adopted. Table 4 outlines the standards used to evaluate the caliber of the researches that were considered.

Based on the quality evaluations, all primary studies (11 articles) were evaluated (Table 4). The goal of each investigation is evaluated in the first question (QA1). In 80% of the investigations, the answer to this question was affirmative. The second question (QA2) evaluated whether or not the study provides a thorough explanation of the methodology. In 67% of the investigations, there was a positive response to this question. The third question (QA3) inquires as to how the outcome will be validated. Only 60% of the research used effective validation strategies. The fourth question (QA4) evaluates whether investigations are grounded in fact rather than opinion or viewpoint. Only 73% of the studies gave a favorable response. The final question (QA5) looks for the total number of citations the research received. As a result, 87% of research received higher citations. Fig. 2 displays the results of the quality assessment.

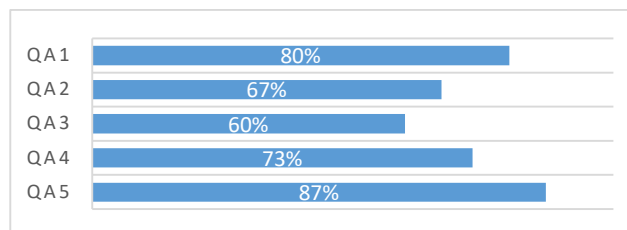


Fig. 2 Quality Assessment Result

### 3) Data Extraction and Synthesis

To gather data relevant to the research subject, data extraction was done. A specified extraction form was used to extract the data (Table 5). We were able to fully document the primary studies that addressed our research question by using this form.

### C. Reporting the Review

By summarizing the studies' findings and responding to each RQ, the review's findings were reported. Based on the outcomes of the data extraction, each RQ's description was created. As a reference guide for concerns, the 2009 PRISMA Checklist used by [16] was accepted.

TABLE 5  
 DATA EXTRACTION FORM

#	Study data	Description	Relevant RQ
1	Identifier	Unique ID for the study	Study overview
2	Title		Study overview
3	Authors		Study overview
4	Year		Study overview
5	Type of article	Journal, conference	Study overview
6	Research goal	What is the study's contribution?	1, 2, 3
7	Background	What is the study's background?	4
8	Research method	What research techniques were used in the study	1, 2, 3
9	Data	What data were used in the study?	1, 2, 3, 4
10	Validation	What method of validation did the study employ?	1, 2, 3
11	Challenge and limitation	Which problems and restrictions did the study acknowledge?	4

## IV. RESULTS

### A. Summary of Studies

We were able to find and analyze 15 primary studies based on the review method. Thirteen (73.3%) were published in journals while four (26.7%) were published in conference. From this result, we can conclude that software security and software testing are discussed in journals frequently.

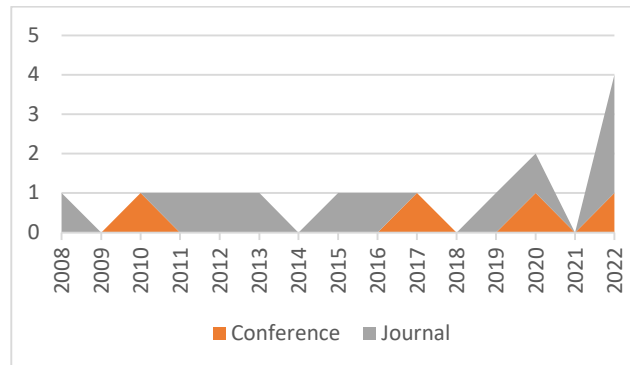


Fig. 3 Study Overview

As for the dates of each study, we found a study as early as 2008 that discussed software security and software study, and the latest were found in 2022, meaning this topic has already been discussed for 15 years, as shown in Fig. 3. However, only journal seems to have consistent amount of study for each year. The highest number of studies published in journals was in 2022 while for conference it was consistent during 2010, 2017, 2020, and 2022. This result indicates that software security and software testing has become quite popular in recent times.

### B. Summary of studies relevant to RQ

We were able to collect 15 papers that satisfied our research questions. Table 6 lists each aspect or strategy's name, along with a brief description.

Duan et al. [13] built a model to evaluate software security based on research into definitions and ideas of software security with relative calculation method. The study is held within web applications and is set as the evaluation object. The proposed method provided a two level hierarchical framework consisted of 25 second-level measures and six first-level metrics. As the result, this study is able to prove that it successfully reflects security level directly.

Hui et al. [1] presented a security defects taxonomy to tackle the issue in software security test. This study used SANS Top 20 attack vectors and MITRE's Common Weakness Enumeration (CWE) top 10 list, so the domain of use would be generally used. As the result, the taxonomy can find software security defects enumeration.

Arnold et al. [20] introduced a testing method to identify any connections between software updates and the likelihood that, when included, they will introduce a security flaw in the software program. They experimented this method using LoDash, Apache Tomcat 8.5, cURL, and Sudo 1.8, which are all used in the web application domain. This study uses Arachni and OWASP for CVE vulnerability assessments. As a result, the technique has demonstrated a link between the volume of code modification and security vulnerability.

TABLE 6  
 STUDIES RELATED TO RQ1

No	Metric/Approach	Author	Description
1	Software Security Evaluation Model	[13]	Evaluation model that is composed of numerous metrics that are connected to one another and have hierarchical relationships. The two-level hierarchical evaluation system that is being presented has 25 second-level metrics and six first-level metrics.
2	Software Security Defect Taxonomy	[1]	A taxonomy that makes an effort to organize data on software security flaws so that, as new flaws are added, testers will have a better understanding of which software system elements and lifecycle stages are more likely than others to produce security flaws, as well as what they should do to fix the problems.
3	Agile Software Development: Assessing Changes to Software Systems' Security Posture to Find Software Security Vulnerabilities	[20]	To determine whether human security testing is required, a novel testing method that assigns system changes a weighted score based on correlations between various types of software changes or the overall amount of code change may be used.
4	EFSMs (Extended Finite State Machine)	[21]	Method for determining whether an EFSM specification is secure utilizing the Java Path Finder verifier and the ESFM Java template
5	Code Defect Analysis	[22]	Technique or framework that scans the software's source code for flaws, security holes, and problems in the quality of the code.
6	Viewnext-UEx preventive secure software development model	[2]	A brand-new, environment-adaptive methodology for developing secure software is suggested.
7	Requirements Dependency Analysis	[23]	An innovative method for integrating horizontal and vertical traceability when examining the connections between security needs.
8	Countermeasure graph for risk analysis	[24]	Procedure combining the threat and risk perspective with the user perspective (misuse and abuse instances) (Peltier and attack trees). The strategy takes into account the objectives of the attacker, the agents, potential attacks from the agents, and attack-prevention measures.
9	Confidence Measures Analysis	[25]	An analysis of software security that measures the degree of alignment between the objective security attribute value and the conclusion of the software security evaluation's dependability level.
10	Goal-oriented Testing	[26]	The study suggests Guider, a brand-new path exploration algorithm. Data dependence analysis is used by Guider to pinpoint the root cause of a test goal execution. To affect how it is executed, it makes use of dynamic symbolic execution-based path exploration. By taking advantage of the program's static control structure, Guider also enhances path discovery. Guider suggested the Framework Sebo, a brand-new goal-oriented testing methodology. To improve the detection of buffer overflow vulnerabilities, Sebo uses symbolic analysis, constraint solving, dynamic program analysis, control and data dependence analysis, and type inference analysis.
11	Threat Modelling	[27]	Threat modeling is a process for comprehending a system's complexity and recognizing all potential dangers to the system.
12	Security Goal Models and Vulnerability Detection Conditions (VDCs)	[28]	Passive testing method that makes use of security goal models, a progression of vulnerability cause graphs, to find software flaws with some automation improvement through the use of formal language.
13	Taxonomy of Security Weakness	[29]	Approach to find out the taxonomy of software security weaknesses through a GitHub commit to fixing bugs.
14	CVE (Common Vulnerabilities and Exposures) Framework	[30]	A comprehensive framework that incorporates a variety of security requirements and enables users to verify the reliability or validity of software security. The 2017 Security Assessment of Corporate Information Systems, the 2017 OWASP Top 10, NIST 800-30, ISO 27034, and ISO 27002 served as the foundation for the framework's parameters. The CVSS was additionally utilized to rank vulnerabilities according to their importance.
15	Dynamic Information Flow Analysis	[31]	An approach that is used to reveal unusual pattern information flow to identify security vulnerabilities in software.

Ermakov et al. [21] suggested a method for determining the security of an Extended Finite State Machine (EFSM) specification in web services domain. An EFSM has predicates that describe the circumstances in which a transition

might occur, input and output parameters, and context variables, in contrast to conventional trace models like Finite State Machines (FSMs). This method uses Java template and verifier Java Path Finder and is proven to be more promising.

Li et al. [22] developed a mechanism for incremental checking that allows for quick source code security audits. To verify the method, they used the data from Juliet Test Suite V1.2 from National Institute of Standards and Technology (NIST). As the result, it is proven that the mechanism is very effective in the ability and accuracy of code defect detection.

Nunez et al. [2] introduced the Viewnext-UEx paradigm, a novel, adaptable, and preventive method for creating secure software to answer demands of new software development methodologies that support the creation of software that is secure by default. The sector of the electric industry was where the experimental undertaking was created to validate the model. As the result, the model was able to reduce 68, 42%, vulnerabilities.

Wang et al. [23] stated that, by concentrating on both the interdependencies between security criteria as well as the individual security requirements, vulnerability detection may be advanced. This study provides a semi-automatic method that integrates horizontal and vertical traceability, and also how the dependencies may be utilized to run security checks and find vulnerabilities. They used five projects in healthcare and education domains to show the significant increase in recall of 81%.

Baca and Petersen [24] handled the issue of low level risks that are reported by developers when using risks analysis in the agile development process. They then offered a fresh approach to risk analysis, which is countermeasure graphs, along with a tool to complement it. A developed countermeasure strategy and tool arose after multiple projects had entered the research, was generalizable enough to function for every project, and was employed even after the study was over.

Ren et al. [25] provided a technique to get software security confidence measures with quantifiable values as the evaluation outcomes. The evaluation conclusion and the related confidence measures may both be obtained using this method's application to small samples of software security evaluation data using the Bayesian theory. Since the study doesn't specify which domains have been tested, we assume that this study can cover general use.

Do et al. [26] investigated and created automated techniques to boost software testing effectiveness. The technique provided to identify probable safety breaches using type inference analysis, and generate test input using dynamic symbolic execution. As the result, the method is able to run in a few seconds and find vulnerability errors, while two other baseline algorithms failed even after 30 minutes.

Bernsmed et al. [27] absorbed knowledge from four different studies in order to create better good practice in using Data Flow Diagrams, STRIDE, and Microsoft Threat Modelling Tool. As the result, this study provides recommendations on how to do threat modelling activities within Agile methodology, so that it can be more useful.

Shahmehri et al. [28] provided a technique for spotting security flaws by looking for signs of their sources in execution traces. This study focuses on using security goal models (SGMs), which models to use to achieve a given goal, and to find vulnerabilities using automated testing. As the result, vulnerability detection conditions (VDCs) are presented.

Mazuero-Rozo et al. [29] introduced the first taxonomy of security flaws in Android apps that is both applicable to Java-related code and Kotlin-related code. After inspecting 681 commits fixing security weakness manually and conducting survey with 43 developers, this study was able to collect 80 types of software security weaknesses.

Naeem et al. [30] analyzed various frameworks and technologies, and then proposed an integrated framework to check the application's validity. It allays the worry of choosing which applications to put on crucial systems. Six PDF readers were used in the study to apply the framework, and it was demonstrated that these programs should not be installed on crucial systems.

Masri et al. [31] introduced a novel method for identifying software security flaws that focuses on making it easier to find and fix security flaws rather than allowing for online attack reaction. It combines anomaly detection with fine-grained dynamic information flow analysis. The method has been applied to four open-source systems and has shown potential to use in general domains.

## V. DISCUSSION

We found that there are 15 studies explaining aspects and approaches to software testing, including software security standards-based approaches to ensure software security credibility and authentication. In addition, there is also an approach that analyzes the most common errors and compiles a taxonomy that can be used as a guideline for software testing. External threats are also considered by looking at the context given to each property or function defined in the program. It also influences the architecture for writing code that crosses inheritance. There is also an integrated framework approach that combines different types of software security standards.



*A. What are the approaches that have been used in software security testing?*

After thoroughly reviewing each study, we were able to group them according to their methodology and substance, as shown in Table 7. We made six categories which are procedural, analysis, taxonomy, metrics, integrated framework, and software development methodology to categorize them. While analysis, taxonomy, metrics, integrated framework, and software development methodology was explicitly stated in some of the studies, we had to create another category for approaches that present steps or procedure, called procedural.

The procedural category holds the greatest number of studies which is nine. This category includes general methods like testing the code or specific criteria such as security requirements [2] [23], information flow [31], or GitHub commit [29]. We group them together so it will be easy to present.

Analysis category is exclusively covered [25] because this study is the only one to analyze a specific item, which is security confidence, and present quantifiable values as the outcome. So, in short, the analysis category covers the kind of study that produces value that reflects something out of specific items in software security.

The third category that we present is taxonomy, which was explicitly mentioned in [1] [29]. Both studies explained how to classify vulnerabilities into some classes, while the next category is metrics, which was also explicitly mentioned in [13]. In this context, metrics is a specific item to test in software security. We believed that taxonomy and metrics deserved to be separated due to two reasons: (1) they are mentioned explicitly and separately, and (2) they are practically different.

TABLE 7  
 APPROACHES BASED ON STUDIES

Category	Studies
Procedural	[20] [21] [22] [23] [24] [26] [27] [28] [31]
Analysis	[25]
Taxonomy	[1] [29]
Metrics	[13]
Integrated Framework	[30]
Software Development Methodology	[2]

Integrated framework is the next category which is explicitly stated in [30]. This category is unique because, in shaping its framework, they used several other frameworks. And the last category is software development methodology [2], which is specifically discussed as to how to develop software to achieve secure by default result. This category explains the whole process of software development and not only in security testing aspect.

As Table 7 shows, procedural is the most used approach in software security testing with nine studies. Taxonomy is the second most used approach with two studies while the other approaches are tied with one study each.

*B. What are the aspects that have been used in software security testing?*

We were also able to identify some aspects or items that are used for software security testing. Some papers used different terms, but we concluded terms that are closely related. For example, ‘information flow’ can be considered as ‘information confidentiality.’ In the end, 14 items were found followed by studies that used them as seen in Table 8.

TABLE 8  
 ASPECTS BASED ON STUDIES

What is tested	Studies
Code	[20] [22] [26] [21] [2] [27] [28]
GitHub commit	[29]
User management	[13]
Access control	[13] [26] [1]
Authentication	[13]
Security Audit	[13]
Information confidentiality	[13] [31] [1] [27]
Data integrity	[13] [26] [1]
Induced causes	[1]
Security requirements	[2] [23]
Design and architecture	[2]
Confidence measures	[25]

The items in Table 8 were derived from studies. Some studies presented aspects and taxonomy [1] [29] [13] that we can derive to specific items to be tested. Other studies mentioned where they would do the software security testing and we collected this information.

The most used item to test is code with seven studies. The second most used item is information confidentiality with four studies. The third most used item is tied between access control and data integrity with three studies. The fourth would be security requirements with two studies. And the other items only have one study each.

As we stated before, because some of the items are derived from taxonomy or aspects, they are usually used together or combined to achieve better result in software security testing. Also, related to the approach in Table 7, Naeem [30] integrated some frameworks to produce a new one, in which it means that these items were also related to the reference that they use to generate, which we will be discussing in point 3.

*C. What are the references that have been used in studies?*

Apart from the approach and aspects, we also spotted the use of other references in software security and software testing such as ISO, OWASP, CVSS, and other. We collected the references and following studies as seen in Table 9.

In this section, we were able to find references related to software security and testing. However, uniquely there are some references related to software development that is used by [2]. These references paid attention to testing to make sure that the security aspect in software is in good quality. Study [2] combined them to create a new methodology in developing software that ensures the security, while the other references like OWASP, NIST, ISO, CVSS, Arachni, SANS, and MITRE's are generally used in software security testing.

TABLE 9  
 REFERENCES USED IN STUDIES

References	Studies
2017 OWASP top 10	[30] [27][20]
NIST 800-30	[30]
ISO 27034	[30]
ISO 27002	[30]
CVSS	[30]
CLASP	[2]
BSIMM	[2]
SAMM	[2]
SDL Microsoft	[2]
TSP SECURE	[2]
OSSA	[2]
Arachni	[20]
SANS Top 20 Attack Sectors	[1]
MITRE's CWE top 10	[1]

From the result we can see that OWASP is the most used reference in the studies with three. OWASP is mostly used in web applications, which implies that web applications are paid more attention in software security testing, which we will discuss in point 4. The other references, however, only receive one study each.

*D. How is software security testing conducted in different domains?*

To understand better in which domains software security testing is applied, we also highlighted the domains of each study. In relation to [6] we also investigated and categorized each domain whether they were considered in safety-critical systems or not. By doing so, we hoped to find a trend in software security testing both in safety-critical systems and in non-safety-critical systems. The results are presented in Table 10.

TABLE 10  
 DOMAINS USED

Domains	is SCS?	Studies
General	No	[1] [22] [24] [26] [27] [28] [31]
Web Applications	No	[13] [20] [21]
Electric Industry	Yes	[2]
Healthcare	Yes	[23]
Education	No	[23]
Android Applications	No	[29]
Critical Systems	Yes	[30]

General domain is the domain that used software security testing the most with seven studies, while web applications came in second with three studies. The other domains tied with only one each.

As we already implied in point 3, OWASP is the most used reference in software security testing, but it turns out that the number of web applications is also high. That would mean there is a correlation between the reference that we used in software security testing and in which domains we deploy the software.

Regarding safety-critical systems, we found out that Electric Industry, Healthcare, and Critical systems are considered SCS. Out of three studies discussing SCS, two of them [2][23] actually test security requirements while the other [30] created an integrated framework. By this finding we conclude that, in SCS, they started software security testing in the early stage, which is in security requirements, and even created an integrated framework to make sure that the security is safe.

TABLE 11  
 APPROACHES AND ASPECTS USED FOR EACH DOMAIN

Domains		General	Web Applications	Electric Industry	Healthcare	Education	Android Applications	Critical Systems
approaches	Procedural Analysis	x	x		x	x		
	Taxonomy Metrics	x	x				x	x
	Integrated Framework							
	Software Development Methodology Code	x	x	x				
aspects	GitHub Commit						x	
	User Management		x					
	Access Control	x	x					
	Authentication		x					
	Security Audit		x					
	Information Confidentiality	x	x					
	Data Integrity	x	x					
	Induced Causes							
	Security Requirements				x	x	x	
	Design And Architecture				x			
Confidence Measure								

In Table 11, we also mapped the strategies and components applied in each domain. For software security testing, studies in the broad field have employed procedural and taxonomic techniques, taking into account elements like code, access control, information confidentiality, and data integrity. Code, user management, access control, authentication, security audit, information confidentiality, and data integrity were among the procedural and metric issues covered in studies in the web applications domain. Only the software development technique, along with security requirements, design, and architecture, were employed in the studies of the electric sector. Studies in the fields of healthcare and education follow a similar pattern that relies on security needs and procedural approaches. Applications for Android have been studied using a taxonomy approach and GitHub commit aspect. Studies on crucial systems, however, only employed taxonomy.

We can get some interesting information from Table 11. Due to the large number of studies and applications, the web applications domain demonstrates the majority of approaches and elements. The web applications domain demonstrates complicated software security testing through the usage of user management, access control, authentication, security audit, information confidentiality, and data integrity in their respective aspects. It is also intriguing to note the similarities in approach and focus between the healthcare and education sectors. As [24] indicates, both domains need to safeguard users, and security needs seem to be the most critical factor to take into account. Integrated framework [31] and confidence measure [26] are two more approaches and aspects that are not employed since they are still viewed as conceptual.

## VI. CONCLUSIONS

From the findings we can conclude that the procedural approach is very often used to test software security testing. The most used test aspect in software security testing is the Code. The most used software security testing reference in a study is OWASP. The high usage of OWASP as a Software Security Testing Reference has a high correlation with Web Application, being the second most often used domain to conduct software security testing. General domain is the most common domain used to conduct software security testing. Although it is the most conducted, general domain doesn't fall into the safety-critical systems categories. This study discovered approaches, aspects, references, and domains that are used in software security and software testing. However, the number of studies used is still very limited. Certain issues such as library or reference that list all possible approach and aspects, best practice combining specific approach and aspects, and research in software security testing as a unit in practical are not yet explored in this research and can be explored in future research.

**Author Contributions:** *Awalurahman*: Conceptualization, Methodology, Writing - Original Draft, Writing - Review & Editing. *Witsqa*: Conceptualization, Writing - Original Draft. *Raharjana*: Conceptualization, Methodology, Review & Editing, Supervision. *Basori*: Writing - Review & Editing.

**Funding:** This research received no specific grant from any funding agency.

**Conflicts of Interest:** *Raharjana* and *Basori* are members of Editorial Teams, but had no role in the decision to publish this article. No other potential conflict of interest relevant to this article was reported.

## REFERENCES

- [1] Z. Hui, S. Huang, B. Hu, and Z. Ren, "A taxonomy of software security defects for SST," *Proc. - 2010 Int. Conf. Intell. Comput. Integr. Syst. ICISS2010*, pp. 99–103, 2010, doi: 10.1109/ICISS.2010.5656736.
- [2] J. C. S. Nunez, A. C. Lindo, and P. G. Rodriguez, "A preventive secure software development model for a software factory: A case study," *IEEE Access*, vol. 8, pp. 77653–77665, 2020, doi: 10.1109/ACCESS.2020.2989113.
- [3] H. Nina, J. A. Pow-Sang, and M. Villavicencio, "Systematic mapping of the literature on secure software development," *IEEE Access*, vol. 9, pp. 36852–36867, 2021, doi: 10.1109/ACCESS.2021.3062388.
- [4] D. Zhang *et al.*, "SimFuzz: test case similarity directed deep fuzzing," *J. Syst. Softw.*, vol. 85, no. 1, pp. 102–111, 2012, doi: 10.1016/J.JSS.2011.07.028.
- [5] R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, "Systematic literature review on security risks and its practices in secure software development," *IEEE Access*, vol. 10, pp. 5456–5481, 2022, doi: 10.1109/ACCESS.2022.3140181.
- [6] J. dos Santos, L. E. G. Martins, V. A. de Santiago Júnior, L. V. Povia, and L. B. R. dos Santos, "Software requirements testing approaches: a systematic literature review," *Requir. Eng.*, vol. 25, no. 3, pp. 317–337, 2020, doi: 10.1007/S00766-019-00325-W/TABLES/11.
- [7] M. M. Hassan, W. Afzal, M. Blom, B. Lindstrom, S. F. Andler, and S. Eldh, "Testability and software robustness: a systematic literature review," *Proc. - 41st Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2015*, pp. 341–348, 2015, doi: 10.1109/SEAA.2015.47.
- [8] M. Aydos, Ç. Aldan, E. Coşkun, and A. Soydan, "Security testing of web applications: a systematic mapping of the literature," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, pp. 6775–6792, 2021, doi: 10.1016/j.jksuci.2021.09.018.
- [9] J. Bozic and F. Wotawa, "Software testing: according to plan!," *Proc. - 2019 IEEE 12th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2019*, pp. 23–31, 2019, doi: 10.1109/ICSTW.2019.00028.
- [10] R. Chamarthi and A. P. Reddy, "Empirical methodology of testing using FMEA and quality metrics," *Proc. Int. Conf. Inven. Res. Comput. Appl. ICIRCA 2018*, pp. 85–90, 2018, doi: 10.1109/ICIRCA.2018.8597290.
- [11] J. D. DeMott, R. J. Enbody, and W. F. Punch, "Systematic bug finding and fault localization enhanced with input data tracking," *Comput. Secur.*, vol. 32, pp. 130–157, 2013, doi: 10.1016/J.COSE.2012.09.015.
- [12] R. Khan, "Secure software development: a prescriptive framework," *Comput. Fraud Secur.*, vol. 2011, no. 8, pp. 12–20, 2011, doi: 10.1016/S1361-3723(11)70083-5.
- [13] Y. Duan, F. Lou, and Y. Fu, *Research of evaluation methods for software security; Research of evaluation methods for software security*. 2016.
- [14] "ISO / IEC 25010 : 2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation ( SQuaRE ) — System and software quality models," 2013, doi: 10.3403/30215101.
- [15] V. V. Ribeiro, D. S. Cruzes, and G. H. Travassos, "Moderator factors of software security and performance verification," *J. Syst. Softw.*, vol. 184, p. 111137, Feb. 2022, doi: 10.1016/J.JSS.2021.111137.
- [16] I. K. Raharjana, D. Siahaan, and C. Faticah, "User stories and natural language processing: a systematic literature review," *IEEE Access*, vol. 9, pp. 53811–53826, 2021, doi: 10.1109/ACCESS.2021.3070606.
- [17] I. K. Raharjana, "A systematic literature review of environmental concerns in smart-cities," *IOP Conf. Ser. Earth Environ. Sci.*, 2019, doi: 10.1088/1755-1315/245/1/012031.
- [18] A. J. Suali *et al.*, "Software quality measurement in software engineering project: A systematic literature review," *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 3, pp. 918–929, 2019.
- [19] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009, doi: 10.1016/J.INFSOF.2008.09.009.
- [20] B. Arnold and Y. Qu, "Detecting software security vulnerability during an agile development by testing the changes to the security posture of software systems," *Proc. - 2020 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2020*, pp. 1743–1748, 2020, doi: 10.1109/CSCI51800.2020.00323.
- [21] A. D. Ermakov, S. A. Prokopenko, and N. V. Yevtushenko, "Checking software security using EFSMs," in *2017 18th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM)*, 2017, pp. 87–90, doi: 10.1109/EDM.2017.7981714.
- [22] X. Li, G. Wang, C. Wang, Y. Qin, and N. Wang, "Software Source code security audit algorithm supporting incremental checking," pp. 53–58, 2022, doi: 10.1109/smartcloud55982.2022.00015.
- [23] W. Wang, F. Dumont, N. Niu, and G. Horton, "Detecting software security vulnerabilities via requirements dependency analysis," *IEEE Trans. Softw. Eng.*, vol. 48, no. 5, pp. 1665–1675, 2022, doi: 10.1109/TSE.2020.3030745.

- [24] D. Baca and K. Petersen, "Countermeasure graphs for software security risk assessment: an action research," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2411–2428, 2013, doi: 10.1016/j.jss.2013.04.023.
- [25] Z. Ren, S. Huang, Y. Yao, and Y. Hong, "Confidence measures analysis of software security evaluation," *Procedia Eng.*, vol. 15, pp. 3505–3510, 2011, doi: 10.1016/J.PROENG.2011.08.656.
- [26] T. A. Do, S. C. Khoo, A. C. M. Fong, R. Pears, and T. T. Quan, "Goal-oriented dynamic test generation," *Inf. Softw. Technol.*, vol. 66, pp. 40–57, 2015, doi: 10.1016/J.INFSOF.2015.05.007.
- [27] K. Bemsmed, D. S. Cruzes, M. G. Jaatun, and M. Iovan, "Adopting threat modelling in agile software development projects," *J. Syst. Softw.*, vol. 183, p. 111090, 2022, doi: 10.1016/J.JSS.2021.111090.
- [28] N. Shahmehri *et al.*, "An advanced approach for modeling and detecting software vulnerabilities," *Inf. Softw. Technol.*, vol. 54, no. 9, pp. 997–1013, 2012, doi: 10.1016/j.infsof.2012.03.004.
- [29] A. Mazuera-Rozo *et al.*, "Taxonomy of security weaknesses in Java and Kotlin Android apps," *J. Syst. Softw.*, vol. 187, p. 111233, 2022, doi: 10.1016/J.JSS.2022.111233.
- [30] R. Z. Naeem, H. Abbas, N. Shafqat, K. Saleem, and W. Iqbal, "A framework to determine applications' authenticity," *Procedia Comput. Sci.*, vol. 155, pp. 268–275, 2019, doi: 10.1016/J.PROCS.2019.08.038.
- [31] W. Masri and A. Podgurski, "Application-based anomaly intrusion detection with dynamic information flow analysis," *Comput. Secur.*, vol. 27, no. 5–6, pp. 176–187, 2008, doi: 10.1016/J.COSE.2008.06.002.

**Publisher's Note:** Publisher stays neutral with regard to jurisdictional claims in published maps and institutional affiliation