





# A Systematic Mapping Study in Security Software Testing for Mobile Devices

Felipe Faustino<sup>1</sup><sup>a</sup>, Jéssyka Vilela<sup>1</sup><sup>b</sup>, Carla Silva<sup>1</sup><sup>c</sup> and Mariana Peixoto<sup>2</sup><sup>d</sup>

<sup>1</sup>*Centro de Informática, Universidade Federal de Pernambuco (UFPE), Av. Jornalista Aníbal Fernandes, s/n, Cidade Universitária, Recife-PE, Brazil*

<sup>2</sup>*Universidade de Pernambuco (UPE), R. Cap. Pedro Rodrigues, s/n, São José, Garanhuns-PE, Brazil*

**Keywords:** Mobile Devices, Security Testing, Systematic Mapping Study.

**Abstract:** Context: Due to mobile devices' popularity, they contain more valuable information. Problem: these devices face many security issues and challenges since smartphones are interesting for security attacks once they contain private and sensitive data. Objective: the aim of this paper is to investigate security testing techniques for mobile devices. Method: a Systematic Mapping Study (SMS) was conducted to identify solutions focused on software security testing for mobile devices. Results: 1264 primary papers were identified, and 17 relevant papers were selected. We found mobile security testing tends to be mostly: dynamic; automated testing; penetration testing; dynamic analysis. Conclusions: dynamic testing represents 58.82% of security testing, followed by static testing, 29.41%, and studies that present both of them 11.76%. It's important to highlight that automated and semi-automated testing represent 88.23% of the studies and only 11.76% used manual testing.

## 1 INTRODUCTION


Mobile devices, such as smartphones, tablets, and wearable devices, have evolved into crucial components for both businesses (Méndez Porrás et al., 2015) and society, playing an indispensable role in the modern world. With a wide range of applications, these devices offer countless functionalities that cater to the needs of consumers. They enable users to accomplish tasks with ease, flexibility, and mobility, rendering traditional computers obsolete in many instances (Oliveira et al., 2019)(Guo et al., 2014)(Russello et al., 2013)(Shezan et al., 2017).


The widespread use of mobile devices has led to an increase in security concerns and challenges. Smartphones, in particular, have become attractive targets for security attacks due to the valuable information they store, including private and sensitive data (Wang, 2015)(Zhong and Xiao, 2014). Malicious applications<sup>1</sup> have emerged as a significant threat,


aiming to steal passwords, track GPS locations, access contacts, and exploit other applications that involve financial transactions or contain sensitive data (Russello et al., 2013)(Avancini and Ceccato, 2013). These security breaches pose a serious threat to the privacy and security of a vast mobile user community (Sun et al., 2016).


Security software testing plays a crucial role in addressing vulnerabilities that compromise information confidentiality, integrity, and privacy (Albuquerque and Nunes, 2016)(Soares et al., 2023). Mobile applications, in particular, present unique challenges compared to traditional web and desktop applications. With their increasing use in critical domains, it becomes essential to adopt specific approaches to ensure application quality and dependability. An effective testing approach is required to assess the high quality and reliability of mobile applications (Zein et al., 2016).

To meet the increasing demand for high-quality mobile applications, developers need to allocate more effort and attention to software development processes. Among these processes, software testing play a crucial role in ensuring the quality of mobile applications. However, one of the challenges related to mobile app testing is the diversity of mobile devices,

<sup>a</sup> <https://orcid.org/0000-0002-2517-5152>

<sup>b</sup> <https://orcid.org/0000-0002-5541-5188>

<sup>c</sup> <https://orcid.org/0000-0002-0597-3851>

<sup>d</sup> <https://orcid.org/0000-0002-5399-4155>

<sup>1</sup><https://www.sciencedirect.com/topics/computer-science/malicious-apps>

known as fragmentation. With multiple OS versions, screen sizes, and resources, testing on a wide range of devices is time-consuming and requires significant human and financial resources, resulting in increased testing costs (Souza et al., 2019).

The findings of Junior et al. (Junior et al., 2022) from their literature review highlight the complexity and importance of security testing. They discovered that security testing is often overlooked compared to other non-functional requirements. Additionally, the use of third-party apps and frameworks can potentially introduce vulnerabilities. Lastly, they observed that apps are not adequately tested against known vulnerabilities.

In this context, there is a need to gain a deeper understanding of *What are the techniques, approaches, methods, tools and processes used for mobile security testing in the literature?*

Our study aims to address this gap by making the following contributions: (1) the identification of techniques, approaches, methods, tools and processes used for mobile security testing in the literature; (2) the classification of these techniques; (3) the criteria and documentation used in the elaboration of those tests; (4) the benefits and gaps in using such techniques, methods, tools or processes and to suggest improvements and prevent mistakes; (5) the current trends and future prospects of mobile security testing; and, (6) the definition of the difference between mobile and non-mobile security testing

Hence, the objective of this study is to conduct a Systematic Mapping Study (SMS) about mobile security testing techniques. Secondary studies have already been used to examine different perspectives of software testing, including: (Garousi and Mäntylä, 2016); automated testing of mobile applications (Méndez Porrás et al., 2015), automated software focused on the Android Platform (Musthafa et al., 2020), and automated functional testing (Tramontana et al., 2019); black, grey, and white box testing approaches (Hamza and Hammad, 2020); usability attributes (Alturki and Gay, 2019), and penetration testing for mobile cloud computing applications (Al-Ahmad et al., 2019). However, there is a gap regarding security testing approaches for mobile devices.

By offering insights into the mobile security software testing approaches, our research outcomes will empower professionals to make informed decisions while selecting the most suitable strategies for conducting security testing on mobile devices present in the testing literature. This will, in turn, contribute to ensuring the overall security, integrity, and reliability of mobile applications in the industry.

This paper is organized as follows: Section 2

presents the background on security software testing and related work. Section 3 presents the research methodology used to perform the systematic mapping study and the context within it is inserted. The results and the analysis related to our research questions are presented in Section 4. Finally, we present our conclusions and future works in Section 5.

## 2 BACKGROUND AND RELATED WORK

In this section, some important concepts related to the field of study on this paper are discussed. Also, the last subsection presents related work.

### 2.1 An Overview of Security Software Testing

Software testing is an integral phase in Software Development Life Cycle (SDLC) process (Dennis et al., 2009), it involves many technical and nontechnical aspects (such as specification, design, implementation, installation, maintenance, and management issues) in Software Engineering (Umar, 2019).

(Myers et al., 2011) define “*Software testing is a process, or a series of processes, designed to make sure computer code does what it was designed to do and that it does not do anything unintended*”.

Regarding software testing classification, in terms of categories (approaches), software testing can be divided into two categories: Static, related to source code only, and Dynamic, related to actual code executions observing its execution (Umar, 2019; Felderer et al., 2016), and in the matter of software testing techniques, considering code analysis, testing is often classified as a black box and white box testing (Copeland, 2004). Tests based on the information about how software has been designed or coded, in which the source code is available and tests are executed based on it, are classified as *white box testing*. On the other hand, enabling mimic hackers attacks, *black box testing* depends on the input/output behavior of the software (Copeland, 2004; Felderer et al., 2016).

Concerning testing types, Umar (Umar, 2019) listed, as a result of a survey conducted by the International Software Testing Qualifications Board (ISTQB), six types as some of the most important types of testing:

- *Functional Testing*. Test functions of the software,
- *Performance Testing*. Testing software responsiveness and stability under a particular workload,

- *Security Testing*. Protect data and maintain software functionality,
- *Usability Testing*. Check ease of use of software,
- *Use case Testing*. Checking that path used by a user is working as intended,
- *Exploratory Testing*. Validate the experience of the user.

Felderer et al. (Felderer et al., 2016) listed confidentiality, integrity, availability, authentication, authorization, and non-repudiation as security properties assets related to security testing. La Polla, Martinelli, and Sgandurra (La Polla et al., 2012) complements that mobility, robust personalization, strong connectivity, technology convergence, and reduced capabilities as the five key aspects that differentiate mobile security from conventional computer security. Felderer et al. (Felderer et al., 2016) additionally classified security testing techniques regarding how they are executed: dynamic or static testing and manual or automated testing. Regarding the techniques' test basis within the secure software development life cycle, Felderer et al. (Felderer et al., 2016) considered four different types:

- *Model-Based Security Testing*. is grounded on requirements and design models created during the analysis and design phase;
- *Code-Based Testing and Static Analysis*. on source and byte code created during development;
- *Penetration Testing and Dynamic Analysis*. on running systems, either in a test or production environment (penetration testing, vulnerability scanning, dynamic taint analysis, and fuzzing);
- *Security Regression Testing*. performed during maintenance.

## 2.2 Related Work

As related work, we search for other reviews about security software testing for mobile devices, aiming to understand what has been done and the possible gaps for this research. We did not find a study with such focus, hence, our discussion of related works is divided in two parts: 1) mobile testing, and, 2) software security testing.

Mobile testing has been studied considering several perspectives such as mobile apps (Souza et al., 2019)(Junior et al., 2022).

Souza et al. (Souza et al., 2019) investigate the use of Exploratory Testing (ET) in mobile apps, aiming to understand its effectiveness and application in a wide range of apps. Two studies were conducted: the first analyzed how testers explore app scenarios

on Google Play, revealing unexplored bugs. The second study applied ET to two apps developed by a software company, identifying bugs of different levels not found using other techniques. Results showed that ET is a promising technique for discovering bugs in mobile apps, but test professionals could learn about the recent results from academia.

Junior et al. (Junior et al., 2022) conducted a SMS to explore dynamic techniques and automation tools for testing mobile apps, particularly focusing on non-functional requirements (NFRs). They found that security is the second most addressed NFR, accounting for 30.3% of the selected studies. An additional analysis of the primary studies revealed that the motivation behind addressing security testing was the lack of attention to security by developers during app implementation. As a result, specific techniques and tools are needed to address this issue. The studies also highlighted that developers often lack sufficient knowledge about security, leading to a lack of vigilance in this aspect.

Web application security testing has also been studied in literature reviews. Seng et al. (Seng et al., 2018) present a systematic review of methodologies and criteria for quantifying the quality of web application security scanners, emphasizing their benefits and weaknesses. It aims to assist practitioners in understanding available methodologies and guiding future development efforts. On the other hand, Aydos et al. (Aydos et al., 2022) present a SMS on web application security testing, addressing research questions, selection criteria, and classification schema. It includes 80 technical articles from 2005 to 2020 and provides an overview of web security testing tools, vulnerability types, and more. The study benefits researchers and developers by offering insights into web application security testing trends and secure development.

Garousi and Mäntylä (Garousi and Mäntylä, 2016) conducted a tertiary study, a systematic literature review of literature reviews in software testing. They identified two relevant secondary studies out of 101 selected between 1994 and 2015, indicating a lack of research on mobile applications. The selected studies focused on automated testing approaches for mobile apps, testing techniques, empirical assessments, and challenges in mobile application testing.

The objective of Zein, Salleh, and Grundy's study (Zein et al., 2016) was to analyze mobile application testing techniques, approaches, and challenges. They conducted a Systematic Mapping Study, mapping and classifying 79 studies into categories such as test automation, usability testing, context-awareness testing, security testing, and general testing. Within

the 79 studies, eight focused on security testing, with inter-application communication threats identified as the main challenge. To provide an updated understanding of mobile security software techniques, this work presents a systematic mapping study to identify methodologies, approaches, and techniques used in mobile security testing.

Regarding the perceptions of practitioners from the mobile software testing environment on security-related testing topics, Soares et al. (Soares et al., 2023) present the results of a survey answered by 49 software testing practitioners, and they concluded that there is also a lack of knowledge about the topics discussed. They also conclude the need to improve security culture. This fact was also observed in the work of (Santos et al., 2021)

### 3 RESEARCH METHODOLOGY

In order to perform this SMS, the guidelines for systematic mapping proposed by Petersen et al. (Petersen et al., 2015) were used.

#### 3.1 Research Question(s)

The purpose of this SMS is to better understand the state-of-the-art regarding mobile security testing and investigate the methodologies, approaches, and techniques used. With these answers, we intend to obtain knowledge about the strong points and gaps of Mobile Security Testing (MST). Therefore, new research can take as a starting point to propose the improvement of the area. Accordingly, we intend to answer the following Research Question (RQs): *What are the techniques, approaches, methods, tools and processes used for mobile security testing in the literature?*

Based on the main RQ, specific questions were raised according to MST aspects that we are interested in. These questions and their motivations are described in Table 1.

#### 3.2 Research String

First, keywords that were used to identify scientific papers in databases were defined. However, to include as many papers as possible, and avoid losses, keywords were defined as broadly as possible. Based on the RQs defined in Table 1, two main keywords were initially identified: Smartphone and Security Testing. In addition, possible variations such as synonyms and regular/plural forms of both of the keywords have been considered, resulting in the following combination search string with the keywords:

Table 1: Research Questions.

ID	Research question	Motivation
RQ1	What are the techniques, approaches, methods, tools and processes used for mobile security testing in the literature?	To get the state-of-the-art in mobile security testing.
RQ1.1	How are the techniques, approaches, methods, tools or processes classified by the authors?	To identify the type of each technique, method, tool or process, such as black box, white box, etc.
RQ1.2	How is the security mobile testing documented by the authors?	To identify the criteria and documentation used in the elaboration of those tests.
RQ1.3	What are the pros and cons of these techniques, approaches, methods, tools or processes to mobile security testing?	To identify the benefits and gaps in using such techniques, methods, tools or processes and to suggest improvements and prevent mistakes.
RQ1.4	What are the current trends of mobile security testing?	To identify the current approaches used in mobile security testing.
RQ1.5	What are the future prospects of mobile security testing (if applicable)?	To understand the path mobile security testing is taking and the future of the field.
RQ1.6	What is the difference between mobile and non-mobile security testing?	To understand the major difference between those tests.

We did not include the term software in the search string because it was not our goal to restrict to software testing. Besides, there are several terms used by the literature in this context such devices, application, system, apps and software as well. Hence, we opted for a wide approach.

#### 3.3 Data Source Selection

After defining the RQs and the Research String, the approach used to perform the extraction of scientific papers was searched based on the research string, presented in previous subsection, in four scientific papers' databases. The following databases were selected: ACM Digital Library, IEEE Xplore Digital Library, Elsevier Science Direct, Springer Link. The searches were conducted using the "Advanced Search" option to each of the respective databases used. Table 2 presents the number of identified primary papers (1264), and all searches were performed on January 13, 2022.

Table 2: Details of the Study Selection Process.

Database	Primary Studies
ACM	249
IEEE	134
Science Direct	547
Springer Link	328
<b>Total</b>	<b>1264</b>



### 3.4 Selection Criteria

During a SMS, the study selection criteria play an important role setting which studies are included or excluded. In addition, the definition of inclusion criteria and exclusion criteria was necessary to allow the impersonal and standardized identification of primary studies by the authors. Therefore, Inclusion Criteria (IC) and Exclusion Criteria (EC) were defined as presented in Table 3.

Table 3: Inclusion and exclusion criteria.

ID	Inclusion Criteria
IC1	Primary studies
IC2	Study published between January 2011 and January 2022
IC3	Full paper accepted in scientific journal or conference
IC4	Studies that report security software testing techniques
ID	Exclusion criteria
EC1	Secondary studies
EC2	Short papers
EC3	Duplicated studies (only one copy of each study was included)
EC4	Non-English written studies
EC5	Studies that do not describe techniques, approaches, methods, tools or processes used for mobile security testing
EC6	Studies that are not available accessed using institutional credentials

A software tool was used to support the SMS protocol definition. The tool, called Parsifal (<https://parsif.al/>), is an online tool designed to support researchers to perform SMS and the results found in the databases were imported to it. Then, the duplicated studies (45) were excluded.

After defining the selection criteria, the first selection step was the identification of relevant papers by applying the IC and EC to the primary studies found in the databases through reading the titles and abstracts. If there was insufficient data, the complete text from the paper was read. At the end of this step, 29 relevant studies were selected as shown in Table 2. Afterward, it was performed a quality assessment in the selected studies (24) and those that do not satisfy a minimum quality score of 50% were excluded (subsection 3.5).

### 3.5 Study Quality Assessment

The quality assessment (QA) of selected studies was achieved by a scoring technique to evaluate the credibility, completeness and relevance of the selected studies. In order to evaluate the selected studies, 8 Quality Assessment Questions (QAQ) were defined and their relation to the RQs is presented in Table 4. The QAQs were answered after the full read of the 24 selected studies.

Table 4: Quality Assessment Questions.

ID	Quality Assessment Questions
QAQ1	Are the techniques, approaches, methods, tools or processes used for mobile security testing well described in the paper?
QAQ2	Is there a classification of techniques, approaches, methods, tools or processes by the authors?
QAQ3	Is there a description of the scope in which the research was carried out?
QAQ4	Are the criteria and documentation well defined?
QAQ5	Are the mobile security tests results validated?
QAQ6	Is there a clear statement of findings?
QAQ7	Is there a description of future work?
QAQ8	Is there a clear statement of threats to validity or studies limitations?

The Rating Score of QA is the result of the sum of the scores for each question, with the Maximum Score (8.0) that the study could achieve after this sum and a Cutoff Score (4.0) in which the study that scored less than 4.0 was rejected. The possible answers for each QAQ were: **yes = 1.0, partially = 0.5 or no = 0.0**. The quality assessment results are presented in this spreadsheet <sup>2</sup>.

After the quality assessment selection, 17 studies were considered in this review. The studies S05, S13, S20, S21 and S26 were excluded due to cutoff grade, and the studies S28 and S29 were not available using our institutional credentials.

### 3.6 Threats to Validity

During the development of this research, some limitations may have affected the obtained results.

**Selection of Relevant Studies.** The set of studies was obtained through a search string presented in subsection 3.4. All existing articles in MST were selected based on the knowledge and experience of the authors. In addition, there's a potential validity bias wherein the papers identified may not necessarily represent methodologies originating from industry but rather academic methodologies. To minimize such bias, inclusion and exclusion criteria were defined, as well as the definition and delimitation of the research's scope. Yet, different authors may have different understandings of these criteria, so the selection results of other authors tend to vary.

**Data Extraction.** Data extraction was performed based on a spreadsheet exported after the first selection using Parsifal containing the primary studies. We use this spreadsheet to collect the data related to the RQs proposed by this SMS. Even though the strategy adopted for extracting the data has helped mitigate threats to the consistency of data extraction, there are

<sup>2</sup>[https://docs.google.com/spreadsheets/d/1hOaagh1l8dSjGb0JEDBvyo1CjEaguG1zZTlh28zJ1\\_c/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1hOaagh1l8dSjGb0JEDBvyo1CjEaguG1zZTlh28zJ1_c/edit?usp=sharing)

still threats of losing some crucial data through the subjective judgment used by the authors during this extraction.

**Conclusion Bias.** A general problem related to publication bias is the tendency of researchers to always publish positive research results instead of negative ones (Kitchenham and Charters, 2007). This risk is low because the objective of this study is to present the state-of-the-art. Moreover, the conclusions presented may have been affected or influenced by the different degrees of knowledge and experience of the authors, as well as by the perspective adopted when analyzing studies. A possible threat to validity is the limited number of papers found, with only 24 papers identified. To extrapolate trends or draw generalizations, a larger sample size would be necessary. Therefore, it is essential to conduct further searches in the future to locate more relevant works.

## 4 RESULTS AND ANALYSIS

This section presents the results and analysis of the RQs introduced in Section 3. The data extraction results can be found in this spreadsheet<sup>3</sup>.

### RQ1 - What Are the Techniques, Approaches, Methods, Tools or Processes Used for Mobile Security Testing in the Literature?

From the selected studies, as shown in Figure 1, the proposals presented in the selected studies are classified considering how the authors themselves refer to them: 5 frameworks (S11, S14, S15, S23, S27); 4 approaches (S01, S08, S19, S24); 4 tools (S03, S04, S12, S19); 2 systems (S10, S16); 2 methods (S22, S25); and 1 prototype (S06). S19 presents not only an approach but also a tool to generate test cases. In addition, Table 5 presents a brief explanation of each selected study.

Based on the selected studies, several techniques, approaches, methods, tools, and processes have been identified. These contributions demonstrate the active research and development efforts in MST, aiming to enhance the security of mobile applications. While it is acknowledged that the distinction between "techniques", "approaches," "methods", "tools" and "processes" might not always be consistently applied across papers, its utility lies in providing a structured

<sup>3</sup>[https://docs.google.com/spreadsheets/d/1hOaaghl18dSjGb0JEDBvvoICjEaguG1zZ1lh28zJ1\\_c/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1hOaaghl18dSjGb0JEDBvvoICjEaguG1zZ1lh28zJ1_c/edit?usp=sharing)

understanding and categorizing within the context of the research facilitating comparative analysis. From this table, we can observe the following key points:

- **Diverse Approaches.** The selected studies cover a wide range of approaches for MST, including automated testing, token authentication, model-based security testing, and biometric tests, among others. This diversity indicates the need for different strategies to address the various security aspects of mobile applications.
- **Tool Availability.** Several tools are mentioned in the table, such as FireDroid, Fuzzino, AndroBugs, SandDroid, Qark, and APSET. These tools serve different purposes, such as enforcing security policies, generating test data, conducting security analysis, and detecting vulnerabilities. Their availability suggests a growing ecosystem of tools specifically designed for MST.
- **Frameworks and Systems.** The table includes frameworks and systems like VAPT Ai, VPDroid, and Metasploit testing framework. These frameworks provide a structured approach and framework for identifying and mitigating vulnerabilities in mobile applications.
- **Prototype and Methods.** The presence of a prototype (SmartDroid) and specific methods (e.g., NFC vulnerabilities and biometric tests with molds) indicates a research focus on exploring novel approaches and techniques for identifying and evaluating security issues in mobile devices.

Overall, the table highlights the diverse range of techniques and tools available for MST, underscoring the importance of employing multiple approaches to ensure the security of mobile applications. The findings provide valuable insights for practitioners and researchers in the field of mobile security testing.

Overall, the findings highlight the ongoing efforts in the MST domain and provide valuable insights for professionals and researchers in selecting appropriate techniques and tools for mobile security testing.

### RQ1.1 - How Are the Techniques, Approaches, Methods, Tools or Processes Classified by the Authors?

In terms of code analysis classification, we observed that most of the studies did not explicitly classify their contributions except study S15, classified as a white box testing and studies S04, S19, and S25 as black box testing.

Moreover, based on Felderer et al. (Felderer et al., 2016) definitions of security testing, Table 6 presents

Table 5: Techniques, approaches, methods, tools or processes used for MST in the literature and a brief explanation.

ID	Type	Brief explanation
S01	Automated testing (approach)	Detection and a checking module to detect the vulnerabilities existing in Android inter-application components.
S03	FireDroid (tool)	An effective security solution for enforcing fine-grained security policies without the need to recompile any internal modules of the Android OS.
S04	Fuzzino (tool)	A test data generator to perform fuzz testing. It enables the tester to perform black-box fuzzing based on a type specification for input values.
S06	SmartDroid (prototype)	An automatic system for revealing UI-based trigger conditions in android applications.
S08	Token authentication (approach)	A grounded theory-based approach to identify common (unit) test cases for token authentication through analysis of 481 JUnit tests exercising Spring Security-based authentication implementations from 53 open-source Java projects.
S10	Automated testing (system)	Based on TAINTDroid, it puts forward and implements the automated testing system, which is applied in the Android emulator and combined with dynamic taint propagation.
S11	VAPTai (framework)	A threat model to detect and mitigate various unknown vulnerabilities in Android & iOS mobile applications and a security testing framework for identification of MitM attacks in mobile applications.
S12	AndroBugs, SandDroid and Qark (tools)	Several applications from the EATL app store (app store from Bangladesh) and tested them with a few well-known testing tools.
S14	AndroBugs (framework)	AndroBugs framework was used as the automated security testing tool for security analysis of mobile money applications on Android using reverse engineering and dynamic analysis tools.
S15	Automated testing (framework)	Employs numerous heuristics and software analysis techniques to intelligently guide the generation of test cases aiming to boost the likelihood of discovering vulnerabilities.
S16	Automated testing (system)	Metasploit testing framework and deployed it on the cloud platform.
S19	Model-based security testing (approach) and APSET (tool)	The approach generates test cases to check whether components are vulnerable to attacks, sent through intents, that expose personal data.
S22	Security vulnerabilities - NFC (method)	Analyze and conducted security testing on NFC-enabled mobile phones based on reader/writer operating mode in a peer-to-peer fashion manner to find a few vulnerabilities.
S23	VPDroid (framework)	Security analysts can customize different device artifacts, such as CPU model, Android ID, and phone number, in a virtual phone without user-level API hooking.
S24	LetterBomb (approach)	Automatically generating exploits for Android apps relying on a combined path-sensitive symbolic execution-based static analysis, and the use of software instrumentation and test oracles.
S25	Biometric tests with molds (method)	Evaluate the security of fingerprint biometric systems embedded in mobile devices.
S27	Vulvet (framework)	Static analysis approaches from different domains of program analysis for detection of a wide range of vulnerabilities in Android apps.

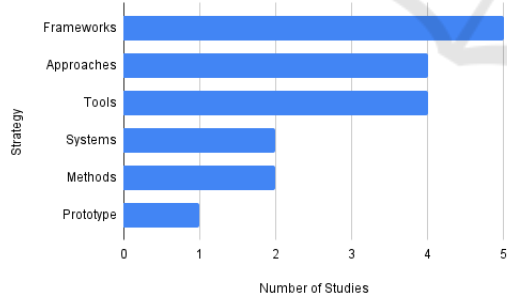


Figure 1: Distribution of number of selected studies by strategy.

how the selected studies are classified according to how the testing technique is executed and Table 7 presents how the techniques are classified according to their test basis within the secure software development life cycle. As shown in Table 6 and Table 7, the predominant categories are Dynamic testing (58.82%) and Automated testing (52.94%), related to the way they are executed, and Penetration testing and dynamic analysis (52.95%), related to their test basis within the secure software development life cycle.

Table 6: Security testing techniques classification according to how they are executed.

Type	Studies	Count	%
Dynamic testing	S03, S04, S10, S12, S14, S15, S16, S19, S22, S23	10	58.82
	S06, S08, S24, S25, S27	5	29.41
Both	S01, S11	2	11.76
Automated testing	S03, S04, S06, S10, S14, S15, S16, S19, S24	9	52.94
Semi-automated testing	S01, S08, S12, S22, S23, S27	6	35.29
Manual testing	S11, S25	2	11.76

From Table 6, we can derive the following conclusions:

- **Dynamic Testing Dominates.** Dynamic testing techniques, which involve the execution of software to identify vulnerabilities, are the most prevalent. They account for 58.82% of the studies included in the table. This indicates the significance of evaluating mobile applications in runtime and assessing their behavior in real-world scenarios.

Table 7: Security testing techniques classification according to their test basis within the secure software development life cycle.

Type	Studies	Count	%
Penetration testing and dynamic analysis	S03, S04, S06, S10, S11, S12, S14, S15, S16, S19, S22, S23, S24, S25, S27	15	88.23
Code-based testing and static analysis	S01, S08	2	11.76
Model-based security testing	-	-	-
Security regression testing	-	-	-

- **Static Testing.** Static testing techniques, which analyze the source code or binary without execution, constitute 29.41% of the studies. These techniques focus on identifying vulnerabilities through code analysis and provide insights into potential security weaknesses.
- **Combined Approaches.** Some studies employ both dynamic and static testing techniques, representing 11.76% of the studies. This approach allows for a comprehensive evaluation of mobile application security by leveraging the strengths of both dynamic and static analysis.

From Table 7, the following conclusions can be drawn:

- **Penetration Testing and Dynamic Analysis.** The majority of studies (88.23%) focus on penetration testing and dynamic analysis techniques. These techniques involve simulating attacks and analyzing the application's response to uncover vulnerabilities. This indicates the importance of actively testing the security of mobile applications during their development and deployment.
- **Code-Based Testing and Static Analysis.** A smaller proportion (11.76%) of the studies utilize code-based testing and static analysis techniques. These techniques involve analyzing the source code or binary to identify security issues. Although less prevalent, they provide valuable insights into the security of mobile applications from a code perspective.
- **Model-Based Security Testing and Security Regression Testing.** These categories have no studies identified in the table, suggesting that they may be less explored or represented in the literature in the context of mobile application security testing.

The tables provide an overview of the classification of security testing techniques based on execution approach and software development life cycle. The findings highlight the prevalence of dynamic testing techniques and penetration testing in the evaluation

of mobile application security. However, the presence of static analysis and code-based testing techniques demonstrates the significance of considering vulnerabilities from a code perspective as well. The absence of studies in certain categories suggests areas for further exploration and research in mobile application security testing.

In summary, the results of this RQ show that we identified a great variety of MST although the lack of explicit classification in terms of white box and black box testing. We also noticed that most of the studies were related to MST in third-party apps, and in this scenario, tests were black box since the tester does not have access to the source code of the application being tested (Myers et al., 2011). Regarding the software development life cycle, model-based security testing and security regression testing were not mentioned. The low rate of Manual testing (11.76%) indicated that researchers are more likely to develop automated or semi-automated testing in their studies.

### RQ1.2 - How Is the Security Mobile Testing Documented by the Authors?

This question aims to make a classification of the selected studies in terms of criteria and documentation used in the elaboration of the described tests. Most studies described the test process or implementation of the proposed tool but did not demonstrate the test process in detail.

Table 8 provides an overview of how the selected studies documented their security mobile testing processes. The table highlights the types of documentation used, such as describing the test process, presenting case studies, identifying test cases, documenting the development process, and providing additional information. It also indicates instances where formal documentation was lacking or not specified.

In study **S01**, 20 applications were examined regarding the effectiveness of their approach and 24 Intents and 3 Content Providers vulnerabilities were found among 195 weaknesses, showing the effectiveness of their automated testing approach. Study **S06** presents several case studies to demonstrate the effectiveness of their prototype.

**S8** identified 53 unique authentication unit test cases, organized by five features (Token Authentication, Token Manipulation, Refresh Token, Login and Logout) and 17 scenarios.

**S10** briefly describes the tests but there is no clear documentation of them. Furthermore, the study reports extra information about the system and apps used during the experiment with a virtual machine and Android version 2.3.3. **S11** analyzed the compliance



Table 8: Classification of Security Mobile Testing Documentation.

Study	Type	Documentation Description
S01	Automated testing	Described the test process and identified vulnerabilities found in the applications.
S06	Prototype	Presented case studies to demonstrate the effectiveness of the prototype.
S08	Token authentication	Identified authentication unit test cases and organized them by features and scenarios.
S10	Automated testing	Briefly described the tests without clear documentation. Provided additional information about the system and apps used.
S11	Compliance analysis	Analyzed the compliance of mobile banking applications to OWASP mobile security risks.
S12, S16, S24	Tools/systems	Tests were performed using the mentioned tools/systems but no further documentation provided.
S14, S04	Development process	Described the development process of the tool implementation and provided case studies for validation.
S22	Security testing	Presented scenarios for security testing and vulnerability attacks.
S23, S25	Strategy applied	Showed the strategy applied without formal documentation.
S15, S19	Automated test case generation	Presented automated test case generation process.
S03, S27	Not specified	No mention of any type of documentation.

of mobile banking applications to the OWASP (Open Web Application Security Project) listed mobile security risks. In **S12**, **S24** and **S16**, the tests were done using the tools/system mentioned, but no further information on how they were performed.

All the process is described by the authors in studies **S14** and **S04**, the authors narrated the development process of the tool implementation and the case studies used to validate the new proposal. Using three devices, LG Nexus 5, Sony Xperia E3 dual and Samsung A5, **S22** presented 2 scenarios. The first, security testing scenario, intended to flood the browser using a single URL opened by the user and the second, vulnerability attacks, sent a few well-crafted packets or messages that take advantage of an existing vulnerability in a previously target phone.

Studies **S23** and **S25** showed the strategy applied but no formal documentation. Test case automated generation was presented in studies **S15** and **S19**. **S03** and **S27** did not mention any type of documents.

The findings from this research question demonstrate the need for further research on software testing documentation beyond MST. While some studies provided detailed documentation of their testing processes, others lacked formal documentation or did not specify any documentation at all. This suggests an opportunity for future investigations to explore and

establish guidelines for documenting security mobile testing effectively.

### RQ1.3 - What Are the Pros and Cons of These Techniques, Approaches, Methods, Tools or Processes to Mobile Security Testing?

The results show that the majority of the studies (58.82%) provided either positive or negative aspects and (17.64% ) provided both positive and negative aspects, as shown in Table 9. Furthermore, there was a significant amount of studies (41,17%) that did not answer this RQ: **S01**, **S08**, **S11**, **S12**, **S19**, **S23**, and **S25**.

Table 9: Advantages and disadvantages of selected studies.

ID	Advantages	Disadvantages
S03	FireDroid is able to monitor any application and system code executed in a device. Making it very effective.	Not Mentioned
S04	It introduced automated security testing based on existing functional test cases using fuzzing heuristics, finding security vulnerabilities more efficiently.	potential weaknesses in the input validation of input type, lengths and value handling of the service.
S06	SmartDroid is very effective in revealing UI-based trigger conditions automatically.	The system was not tested by a large amount of Android applications.
S10	efficiency, avoid undetected controls or Activities in human interaction.	Android's system modification.
S14	tool accessibility	Not Mentioned
S15	automated test case generation, code coverage and uncovers potential security defects, highly scalable.	Not Mentioned
S16	Accurate calculate the number of vulnerable apps. Detect all vulnerability levels, complete basic functional requirements, and simulate real network environment.	Not Mentioned
S22	Not Mentioned	NFC-enabled mobile phones still have weakness in security issues.
S24	Reduction of false positives produced by security analysis and time a human analyst must spend examining a vulnerability.	Not Mentioned
S27	Better result in false positive.	Cannot cover all possible vulnerabilities.

FireDroid (S03). The main advantage highlighted over other Android security approaches is the ability to monitor any application and system code running

on a device. This makes FireDroid a powerful tool for detecting potential security breaches.

Fuzzino (S04). Fuzzino is a test data generation tool used to perform fuzzing tests on applications. This approach aims to insert invalid, unexpected, or malicious input into a program to identify potential vulnerabilities or security holes.

SmartDroid (S06). SmartDroid is effective in automatically revealing UI-based trigger conditions, which is valuable for detecting security vulnerabilities in Android applications. However, further testing at scale is needed to validate its effectiveness in different scenarios.

TAINTDroid (S10). Combining TAINTDroid with the Android emulator allows you to simulate various scenarios and interactions, allowing for an in-depth analysis of mobile application security. However, the Android system modification is mentioned as a disadvantage.

AndroBugs (S14). The main advantage is the accessibility of the AndroBugs framework, which allows security analysts to identify vulnerabilities in mobile applications. However, potential limitations are not explicitly mentioned.

Automated Testing Approach (S15). The main advantage is the automated generation of test cases, combining intelligent heuristics and software analysis techniques. However, the automated approach may not cover all scenarios and vulnerabilities.

Tao et al. (S16). The main advantage is the ability to detect vulnerabilities at different vulnerability levels. However, effectiveness may depend on testers' experience.

Fahrianto et al. (S22). The study analyzes the security of NFC devices, highlighting weaknesses in certain modes of operation.

Garcia et al. (S24). The ability to identify vulnerabilities and determine whether they are exploitable is a positive advantage. The approach reduces false positives and the time required for human analysis.

Gajrani et al. (S27). The multi-tier approach presents better results in relation to false positives, but it still cannot cover all possible vulnerabilities.

### RQ1.4 - What Are the Current Trends of Mobile Security Testing?

The selected studies were published between 2012 and 2021 as shown in Figure 2. The higher number of publications occurred in 2017 and 2020 with three studies each (2017 - S11, S12, S24 and 2020 - S04, S08, S27) followed by 2012, 2013, 2014 and 2016 with two studies (2012 - S06, S15; 2013 - S03, S19; 2014 - S01, S10; and 2016 - S22, S25). The least

amount occurred in 2015, 2018 and 2021 with only one study (2015 - S16; 2018 - S14; and 2021 - S23) and in 2019 no studies were found.

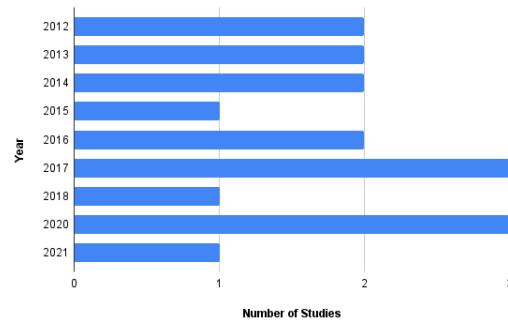


Figure 2: Distribution of number of selected studies by year.

Although the apparent constancy of studies number on MST, it is hard to say that there is a trend in the topic raised by this SMS. As presented in RQ1.1, most of the selected studies are related to dynamic, automated, and penetration testing and dynamic analysis. This result may indicate the benefits of automated testing, once compared to manual testing, the former is more competitive (Tao et al., 2015).

In Figure 3, we present the distribution of the research about MST over the years.



Figure 3: Trend in MST Research.

### RQ1.5 - What Are the Future Prospects of Mobile Security Testing (if Applicable)?

Most of the studies propose improvements for their works (70.58%), indicating the gaps and the need of complementary studies, except studies S08, S10, S12, S23 and S24.

Table 10 summarizes the future directions and proposed improvements in MST studies. Each study (identified by the study code) suggests specific areas of improvement, providing insights into the ongoing research efforts in the field of mobile security testing.

The results of this research question are important to verify the need for complementary studies regard-

Table 10: Future Directions and Proposed Improvements in MST Studies.

Study	Future Directions/Proposed Improvements
S01	Explore automatic analysis of results and discover additional weakness types and attack behaviors.
S03	Improve the developed tool by incorporating dynamic analysis techniques like information flow and taint analysis.
S04	Enhance Fuzzino by adding more fuzzing heuristics and cross-site scripting vulnerability detection.
S11	Extend the approach to other mobile platforms, such as and BlackBerry, Windows, and iOS.
S14	
S15	Improve test case generation techniques and create a graphical reporting environment.
S16	Extend the range of security tests by including more tools and expanding the scope of testing.
S19	Extend the generation of partial specifications and integrate them to test composite components.

ing MST and add new scenarios of testing, including other platforms, improvements of what was developed, and the inclusion of automated analysis.

### RQ1.6 - What Is the Difference Between Mobile and non-mobile Security Testing?

The purpose of this research question was to understand the significant difference between mobile and non-mobile security testing. From the selected studies, only study S01 mentioned the topic in related work pointing out that non-mobile tests as traditional testing techniques and applied to desktop and web applications. We infer that our research string, including mobile in its terms, prevents us from finding more definitions in this context.

## 5 CONCLUSIONS AND FUTURE WORK

In this work, we conducted an SMS to understand the state-of-the-art mobile security testing techniques better and investigate the strategies used. We identified as contributions frameworks, approaches, tools, systems, methods and prototypes, showing the range of options for MST. We also identified that most of these contributions did not explicitly classify their studies related to code analysis, classifying one of them as white box and three studies as black-box testing. In addition, dynamic testing represents 58.82% of security testing according to how they are executed, followed by static testing, 29.41%, and studies that present both of them 11.76%. It's important to highlight that automated and semi-automated testing represent 88.23% of the studies and only 11.76% used manual testing and we infer this as a tendency in MST.

According to the techniques' test basis with the secure software development lyfe cycle, penetration testing and dynamic analysis (52.95%) and Code-based security testing and static analysis (11.76%) were the only types found, excluding model-based security testing and security regression testing as presented in RQ1.1, indicating a research gap and the trends of this field. To summarize, MST tends to be mostly dynamic, automated testing and penetration testing and dynamic analysis.

Among 17 identified studies, we identified a great diversity of MST documents but also the absence of formal documentation in most of the studies, suggesting the need for additional research that elucidates software testing documentations beyond MST. In addition, some of the studies did not provide the advantages and disadvantages of applying the developed contribution and as future work, they suggest the inclusion of other platforms, improvement of what was developed and the inclusion of automated analysis.

The results presented in this SMS can be very useful to testers and researchers since it gathers the current state-of-the-art regarding mobile security testing techniques and indicates its prospects. As future work, we intend to continue this systematic mapping study to explore how security mobile testing is performed in the industrial context, gathering the results found with those techniques by creating a catalog with good MST practices.

## REFERENCES

- Al-Ahmad, A. S., Kahtan, H., Hujainah, F., and Jalab, H. A. (2019). Systematic literature review on penetration testing for mobile cloud computing applications. *IEEE Access*, 7:173524–173540.
- Albuquerque, A. B. and Nunes, F. J. B. (2016). A proposal for systematic mapping study of software security testing, verification and validation. *International Journal of Computer and Information Engineering*, 10(6):1038–1044.
- Alturki, R. and Gay, V. (2019). Usability attributes for mobile applications: a systematic review. *Recent trends and advances in wireless and IoT-enabled networks*, pages 53–62.
- Avancini, A. and Ceccato, M. (2013). Security testing of the communication among android applications. In *2013 8th International Workshop on Automation of Software Test (AST)*, pages 57–63.
- Aydos, M., Aldan, Ç., Coşkun, E., and Soydan, A. (2022). Security testing of web applications: A systematic mapping of the literature. *Journal of King Saud University-Computer and Information Sciences*, 34(9):6775–6792.
- Copeland, L. (2004). *A practitioner's guide to software test design*. Artech House.

- Dennis, A., Wixom, B., and Tegarden, D. (2009). Systems analysis and design with oop approach with uml 2.0, 4th editio.
- Felderer, M., Büchler, M., Johns, M., Brucker, A. D., Breu, R., and Pretschner, A. (2016). Security testing: A survey. In *Advances in Computers*, volume 101, pages 1–51. Elsevier.
- Garousi, V. and Mäntylä, M. V. (2016). A systematic literature review of literature reviews in software testing. *Information and Software Technology*, 80:195–216.
- Guo, C., Xu, J., Yang, H., Zeng, Y., and Xing, S. (2014). An automated testing approach for inter-application security in android. In *Proceedings of the 9th International Workshop on Automation of Software Test, AST 2014*, page 8–14, New York, NY, USA. Association for Computing Machinery.
- Hamza, Z. and Hammad, M. (2020). Testing approaches for web and mobile applications: An overview. *International Journal of Computing and Digital Systems*, 9(4):657–664.
- Junior, M. C., Amalfitano, D., Garces, L., Fasolino, A. R., Andrade, S. A., and Delamaro, M. (2022). Dynamic testing techniques of non-functional requirements in mobile apps: A systematic mapping study. *ACM Computing Surveys (CSUR)*, 54(10s):1–38.
- Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.
- La Polla, M., Martinelli, F., and Sgandurra, D. (2012). A survey on security for mobile devices. *IEEE communications surveys & tutorials*, 15(1):446–471.
- Méndez Porras, A., Quesada López, C. U., and Jenkins Coronas, M. (2015). Automated testing of mobile applications: A systematic map and review.
- Musthafa, F. N., Mansur, S., and Wibawanto, A. (2020). Automated software testing on mobile applications: A review with special focus on android platform. In *2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 292–293. IEEE.
- Myers, G. J., Sandler, C., and Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.
- Oliveira, M. d., Seabra, R. D., and Mattedi, A. P. (2019). Usabilidade de aplicativos de segurança colaborativa para smartphones: uma revisão sistemática. *Revista de Sistemas e Computação-RSC*, 8(2).
- Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and software technology*, 64:1–18.
- Russello, G., Jimenez, A. B., Naderi, H., and van der Mark, W. (2013). Firedroid: Hardening security in almost-stock android. In *Proceedings of the 29th Annual Computer Security Applications Conference, ACSAC '13*, page 319–328, New York, NY, USA. Association for Computing Machinery.
- Santos, P., Peixoto, M., and Vilela, J. (2021). Understanding the information security culture of organizations: Results of a survey. In *XVII Brazilian Symposium on Information Systems*, pages 1–8.
- Seng, L. K., Ithnin, N., and Said, S. Z. M. (2018). The approaches to quantify web application security scanners quality: A review. *International Journal of Advanced Computer Research*, 8(38):285–312.
- Shezan, F. H., Afroze, S. F., and Iqbal, A. (2017). Vulnerability detection in recent android apps: An empirical study. In *2017 International Conference on Networking, Systems and Security (NSysS)*, pages 55–63.
- Soares, A., Vilela, J., Peixoto, M., Santos, D., and Silva, C. (2023). Perceptions of practitioners on security-related software testing in a mobile software development company. In *Proceedings of the XIX Brazilian Symposium on Information Systems*, pages 348–355.
- Souza, M., Villanes, I. K., Dias-Neto, A. C., and Endo, A. T. (2019). On the exploratory testing of mobile apps. In *Proceedings of the IV Brazilian Symposium on Systematic and Automated Software Testing*, pages 42–51.
- Sun, M., Wei, T., and Lui, J. C. (2016). Taintart: A practical multi-level information-flow tracking system for android runtime. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16*, page 331–342, New York, NY, USA. Association for Computing Machinery.
- Tao, D., Lin, Z., and Lu, C. (2015). Cloud platform based automated security testing system for mobile internet. *Tsinghua Science & Technology*, 20:537–544.
- Tramontana, P., Amalfitano, D., Amatucci, N., and Fasolino, A. R. (2019). Automated functional testing of mobile applications: a systematic mapping study. *Software Quality Journal*, 27:149–201.
- Umar, M. A. (2019). Comprehensive study of software testing: Categories, levels, techniques, and types. *International Journal of Advance Research, Ideas and Innovations in Technology*, 5(6):32–40.
- Wang, Y. (2015). An automated virtual security testing platform for android mobile apps. In *2015 First Conference on Mobile and Secure Services (MOBISERV)*, pages 1–2.
- Zein, S., Salleh, N., and Grundy, J. (2016). A systematic mapping study of mobile application testing techniques. *Journal of Systems and Software*, 117:334–356.
- Zhong, H. and Xiao, J. (2014). Design for a cloud-based hybrid android application security assessment framework. In *2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS)*, pages 539–546.