# Security Testing in The Wild: An Interview Study

Dario Di Dario, Valeria Pontillo, Stefano Lambiase, Filomena Ferrucci, Fabio Palomba
Software Engineering (SeSa) Lab – University of Salerno, Salerno, Italy
ddidario@unisa.it, vpontillo@unisa.it, slambiase@unisa.it, fferrucci@unisa.it, fpalomba@unisa.it

***Abstract*—Modern software systems are increasingly complex and the risk of falling into security concerns is high if these systems are not developed with a proper security mindset. Despite the empirical studies and security-oriented approaches proposed by researchers and tool vendors, we still point out a lack of knowledge on the security testing *processes* applied by companies to reduce risks connected to software security. In this paper, we aim to bridge this gap of knowledge by performing an interview-based study with 19 security experts to understand how companies arrange security testing and how the process of security testing is actually performed in practice. Our results highlight that some companies incorporated the figure of the security tester in the software life cycle, yet practitioners reported a lack of standardized guidelines for security testing. From a management perspective, our results suggest that the introduction of formal communication between development and security testing teams may lead to better performance.**

***Index Terms*—Security Testing; Software Vulnerability; Project Management; Software Organizational Structures.**

## I. Introduction

The complexity of modern information technology (IT) systems is rapidly increasing, also due to the rise of novel technologies like cloud computing, microservices, and artificial intelligence, together with sensitive data processing and the permanent connection to other systems [13]. Software security still represents a key concern for practitioners, who are called to design and verify software systems with a security mindset to avoid incidents that may have catastrophic consequences on both the technical infrastructure and the surrounding environment [26, 40]. Software vulnerabilities, security penetrations, and cybersecurity issues have, for instance, frequently led malicious attackers to steal sensitive information [27, 28, 38] and cause financial loss, reputational damage, and legal problems for individuals and companies [23, 48], especially in the context of safety-critical domains [1, 29].

Software testing, i.e., the set of software engineering practices through which practitioners assess the quality and reliability of the software systems being developed [32], plays therefore a major role to ensure software trustworthiness. However, unlike traditional software testing, security testing has different goals and makes use of different instruments, being focused on the software system from the perspective of a malicious user [33]. Nevertheless, practitioners should design for security since the inception of the project and take care of the evolution of possible security concerns over the subsequent development and verification activities [35]: this is the core concept that led to the so-called *"security-by-design"* [25].

Over the last decades, the research community has been supporting practitioners under multiple technical perspectives, contributing to the design of novel secure programming practices [12, 17], software vulnerability detection methods [36], penetration testing techniques [30], other than with the definition of multiple security-oriented lifecycle models [39, 44] and security frameworks, e.g., *Microsoft Secure Development Life-cycle* [19] and *SAMM* from OWASP.[1]

Recognizing the current body of knowledge, we highlight a notable lack of insights into the *processes* currently applied by companies when approaching security testing. Indeed, as further elaborated in Section II, previous studies mainly focused on the frameworks and tools enabling security testing while not focusing on the managerial and operating efforts that practitioners currently put in place nor the practitioners' needs with respect to how the available frameworks and tools should be employed throughout the software lifecycle. This paper aims to advance the current state of the art by conducting a qualitative investigation into the managers' and practitioners' perceptions of security testing. First, we perform structured interviews—involving 19 experts from three large-scale private IT companies—to gather an overview of the security practices and methods employed in their own working environments. Secondly, we perform a semi-structured interview with one of those experts, i.e., a *Chief Technology Officer* (CTO) of a major Italian IT software company having more than 20 years of experience in developing secure software, to get further insights on the current managerial and operating practices and methods employed to face software security concerns.

The key findings of the study show that some companies use security frameworks, varying them according to the application to be developed. While all the involved practitioners recognized the need for demanding security testing practices from specific, experienced teams, security testing is still far from being a standard practice and, indeed, companies often end up with problematic conditions, e.g., dissemination of personal data. Perhaps more importantly, security testing represents a managerial concern: in most cases, this is not done consistently throughout the software lifecycle and the lack of collaborations between development and security units further increases the chance of security issues. We conclude our paper by distilling lessons learned and insights for further research on the matter.

## II. Related Work

Security testing aims at identifying potential security weaknesses in the development process before attackers can exploit them. To integrate security testing aspects into the software

[1]The OWASP SAMM: https://owasp.org/www-project-samm/

lifecycle process, the so-called *Secure Software Development Lifecyle* (SSDLC) arose [31]. In particular, MICROSOFT and OWASP developed frameworks based on SSDLC [47]. The former, proposed in 2004 MICROSOFT's Security Development Lifecycle (MS-SDL) [34], is a framework that integrates security into the software development process at every stage, from design to deployment, while the latter, introduced in 2009 OPENSAMM [8], is a framework that helps practitioners determine which secure application development program components need more attention during development.[2] The employment of these frameworks has been studied by Khan et al. [22] in the context of a systematic mapping study of secure development approaches. Their result showed that the two frameworks are the most widely used by companies.

Other researchers focused on new paradigms to include security aspects in the development processes. DEVSECOPS (Development, Security, Operations) aims to introduce security features in DEVOPS, i.e., a development process that includes a set of continuous practices [41], increasing communication, collaboration, and integration between the development/operations teams and the security team.

Recently, Rajapakse et al. [37] uncovered the main challenges in transitioning to DEVSECOPS. While security process automation tools are necessary, the authors pointed out that considering human aspects is equally important to properly handle security issues: in particular, understanding when and why security countermeasures should be put in place is essential. Our work is motivated by the study by Rajapakse et al. [37]: we indeed aim to understand how security practices are currently applied throughout the software lifecycle.

Thompson [43], and Potter et al. [35] highlighted that developers must be proactive toward security testing and integrate it into the development lifecycle. With respect to these works, we aim to advance the state of the art through the investigation of how developers tackle security risks and how they employ security frameworks to address misinformation about testing and communication failure. Furthermore, we evaluate the effectiveness of these frameworks in simplifying security processes during software development.

One notable aspect that Kreeger et al. [24] has highlighted is the need for more knowledge in security testing. To be an effective security tester, one must possess transversal competencies essential for this field. These competencies include having high-level programming skills, the ability to develop tests effectively, and expertise in security cryptography. In this respect, we aim to reduce the gap by analyzing the background required by companies, doing a first step toward understanding the expertise currently required to perform security testers, other than rising the awareness of researchers and practitioners with respect to the roles involved in security testing practices.

All in all, our work advances the state of the art by providing two main contributions. First, stemming from previous literature showing the need for understanding the processes involved in security management, we aim at investigating the current managerial and operating practices applied within companies when dealing with software security. Second, our work builds on the literature reporting on the need of profiling software security experts [24] to provide insights into the professional roles involved in a typical software security testing process, other than the organizational structure required to effectively deal with software security.

[2]The OWASP SAMM: https://owasp.org/www-project-samm/

## III. RESEARCH STUDY DESIGN

The *goal* of the study was to enlarge our knowledge of the security testing activities performed in industry, with the *purpose* of providing researchers and practitioners with insights into the processes put in place and the possible limitations that would be instrumental to identify further improvements. The *perspective* is both of researchers and practitioners: the former are interested in overviewing the current state of the practice and possibly identifying limitations that may lead to the definition of novel, improved instruments to support practitioners; the latter are interested in understanding the practices applied by fellow practitioners, possibly identifying further managerial tools to put in place in their own contexts.

Specifically, our research agenda targeted two main objectives. First, we aimed at eliciting initial insights into how security testing is arranged in practice, namely whether and how companies organize security testing teams. This research angle is key for contextualizing the role of the security tester, possibly identifying limitations in terms of awareness and managerial aspects of software security. Hence, we asked the following research question:

> **$RQ_1$:** *How is security testing arranged in the industry?*

Afterward, we focused on the current processes applied by practitioners throughout the software development lifecycle. With this research question, we aimed to provide an overview of how security testing is treated over time, in an effort to highlight the current challenges faced by practitioners — and, therefore, the possible additional support requested by researchers. In particular, we asked:

> **$RQ_2$:** *How are security testing activities performed during software development?*

To answer our research questions, we designed a qualitative investigation featuring a combination of structured and semi-structured interviews involving IT practitioners. Figure 1 overviews our research method. The first interviews aimed to provide an overview of security testing in practice, while through the semi-structured interview, we could deepen some aspects and provide a more critical discussion on the matter.

### A. Step 1: Data Gathering

As a first step of our study, we performed structured interviews with 19 practitioners from three companies' experts in security testing activities. Moreover, we conducted an additional semi-structured interview with one of these participants,

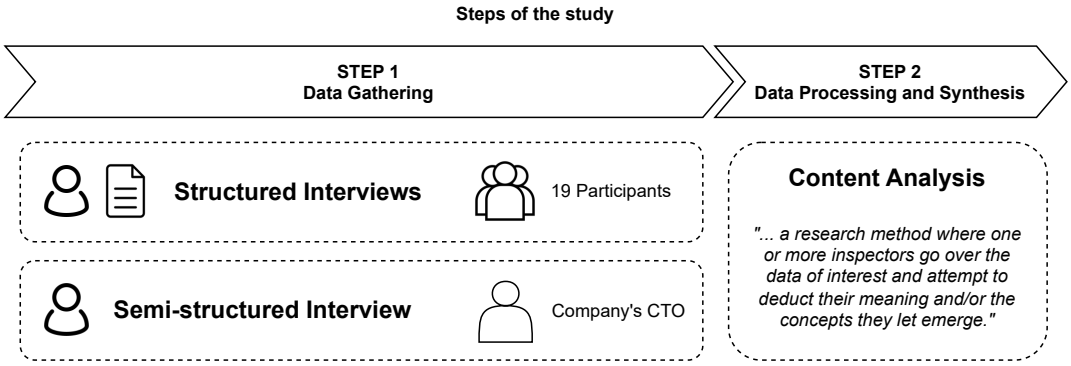


Fig. 1. Overview of our research design.

i.e., the CTO of one of the companies. In the following, we present the design of our interviews.

*1) Design of the Structured Interviews:* For the design of our structured interviews, we relied on the guidelines provided by Andrews et al. [3], Hove and Ada [18], and Stol et al. [42]. Specifically, we based our work on the following principles:

- We asked participants to respond to our question using predefined multiple-choice or scales (numerical or qualitative) answers to give a more analytical character to the data;
- We made sure to use a clear, unambiguous, and coincidental vocabulary to eliminate doubt about the meaning of questions and possible answers;
- We started our interviews with a clear statement of our objectives, and we made clear from the outset that the information taken would not be used for any other purpose;
- We guaranteed respect for privacy by specifying that the data would have been treated anonymously from the point of view of analyzing and publishing the results without referencing who provided the responses.

Based on the previously-mentioned principles, the structured interview was formalized into four consecutive sections: two for the study goal and two for the respondent's information gathering. Specifically, the survey was structured as follows: (1) the first section aimed to provide a brief description of the research and definitions of security testing; (2) the second section was concerned with the use of security testing in real business contexts; (3) the last section deals with business and personal information. For the sake of space, we reported the entire list of questions in our online appendix [11].

Regarding the first section, we wanted to provide a clear definition—reported in the box below—of security testing to avoid ambiguities or incomprehension.

**Security Testing Definition**

*Security Testing can be defined as the set of activities aimed at predicting and finding potential security problems in IT systems, with the goal of increasing product resilience to malicious attacks [13, 35].*

The second section represented the core part of our interview. It contained various questions to elicit (1) who performs security testing activities and the degree of awareness that they have and (2) if such activities follow a rigid and standardized approach. We can separate them as follows:

1) **Subgroups**: As a first question, we asked whether the company has subdivisions according to the activities to be performed. The question might seem trivial, considering that most companies are now divided into rigid subgroups—e.g., developers, testers, and security specialists. However, the question may come in handy to understand whether the low importance of security aspects may also depend on the wrong composition of work activities.
2) **Who performs security testing**: As a second question, we asked participants if, in the context of their company, security testing activities are performed by specific teams or individuals with different roles and selected for their skills. Indeed, we aimed to gather knowledge to answer our first research question, i.e., who performs security testing.
3) **Secure Development Life Cycle and Framework**: Participants are asked if they are aware of the secure development life-cycle, but mainly if they apply frameworks such as the one created by Microsoft (MS-SDL) or any other available framework.
4) **Security Management**: Following the software life cycle, users are asked to specify at what stage of the life cycle testing activities are most often applied. In this manner, we are trying to understand at which stage companies intend to spend more energy. Furthermore, it may be helpful to understand whether the frameworks specified earlier are fully executed or partially.
5) **Testing Methodologies**: By analyzing the responses, it could assume the possible techniques applied. Participants could list the techniques used or write a high-level description—e.g., "*requirements analysis plays a crucial role in security*" is likely that "Architecture security review" techniques are used.
6) **Types of Applications Developed**: Such a question was helpful to understand whether frameworks or unique techniques applied to the life cycle are changed due to the type of application to be developed.

7) **Security Testing in Groups**: Such a question aimed at understanding whether security testing activities are performed in groups.

As the reader may observe, the questions cover multiple angles of security testing, hence allowing us to understand various aspects of how it is performed. In addition, the questions aimed at gathering insights that might be exploited by researchers to further elaborate on the matter.

The last two parts of the interview refer to the collection of company and participant characteristics. The company characteristics help compare and get a complete picture of the companies that participated in the survey. The participant details help to obtain more information about the answers given if needed, for getting them to participate in future surveys such as focus groups or one-on-one interviews, and for sharing the survey results with them if they are interested.

Regarding the participants' selection, we adopt a *convenience sample* approach, i.e., a non-probabilistic sampling method where the sample is taken based on proximity, availability at a given time, or willingness to participate in the research [16]. Specifically, we contact 19 security experts from three companies belonging to our contact network of private businesses. Such a strategy is known to be particularly useful in obtaining participants rapidly but suffers in terms of the generalizability of findings [5, 16]. Nevertheless, we decided to adopt it to address a critical challenge: security experts represent a minority in the software development audience of professional roles—recent statistics reveal that there are only 32,349 experts worldwide and that medium-size company have few security experts (around 13%).[3] As such, we could not run large-scale experimentation, as finding security experts would have been challenging.

*2) Design of the Semi-structured Interviews:* In order to obtain more general data on the topic of security testing, we conducted a semi-structured interview with one of the *Chief Technology Officer* (CTO) of the three companies—as an expert in software engineering and secure software development. In the context of our study, a single interview enables a joint discussion on current security testing practices based on the software development life-cycle.

To design our semi-structured interview, we relied on the guidelines provided by Hove and Ada [18]—more details about the questions are in our online appendix [11]. During the meeting, the first author briefly explained the research methods used and gave a very brief presentation of the current definition of security testing and the life cycle of secure development, with several existing frameworks in the literature. The second and third authors participated in the interview to transcribe information and support the first author. The entire interview was recorded—with permission of the interviewee—and at the end, the authors combined the information with the manual transcription of the entire interview.

[3]U.S.A. statistics on Security Experts: https://www.zippia.com/security-specialist-jobs/demographics/

### B. Step 2: Data Processing and Synthesis

Once we had collected the responses, we performed a quality assessment phase. In particular, the paper's first author reviewed the individual responses collected to filter them. In the end, all the interview responses have been considered acceptable. To analyze the responses, we performed *qualitative content analysis* [7], i.e., a research method where one or more inspectors go over the data of interest and attempt to induct their meaning and/or the concepts they let emerge. Specifically, the process consisted of two steps: (1) iteratively develop the codebook and (2) apply the codes to the raw data. The process was conducted by the first three authors of the paper, who jointly analyzed the individual responses to develop the initial codebook. After developing a unique set of codes, the first author applied them to the rest of the data and performed an ulterior step of coding that resulted in theoretical saturation. We did not rely on any support tool—e.g., *NVivo* and *Atlas.ti*—for performing the coding phase.

To exemplify the data analysis procedure applied, let us discuss the following case, which presents an excerpt of how we passed from raw data to codes:

**Raw Data:** *"We ask companies such as OWASP Italia, Ernst & Young, Deloitte, and others to conduct further security testing. Given the interaction between three different teams, we found it necessary to manage communications through two managerial figures who communicate..."*

**Codes:** *Contacting external stakeholders; Performing Security Testing; Involving more managerial figures; Managing communication between teams.*

All the codes were extracted from raw data using an inductive approach. The only exception is represented by the actions conducted to identify the phases in which security testing is performed: in this case, we applied a deductive approach to map raw data onto the development phases, e.g., software requirements, specification, design, and so on. Overall, we used 22 codes: 1 general (i.e., "Performing Security Testing"), 6 for $\mathbf{RQ_1}$, 15 for $\mathbf{RQ_2}$. All the codes are reported in our online appendix [11]. Three of the codes mentioned above (i.e., *External Actor*, *Contacting external teams*, and *Involving more managerial figures*) were extracted by the semi-structured interview alone. After creating the codes, we analyzed them and their frequency to extract insight and build our findings.

## IV. Analysis of the Results

For the sake of comprehensibility, we split the section into three subsections, one for the participants' general information and two for each research question. Due to space limitations, the raw results are available in the online appendix [11].

### A. Participants Background Information

Regarding the general information on the companies involved in the study, it is worth noting that all of them are multinational IT security and development companies with at least 200 employees and multiple corporate offices worldwide. Furthermore, our interviewees are involved in software projects developed in teams composed of at least six team

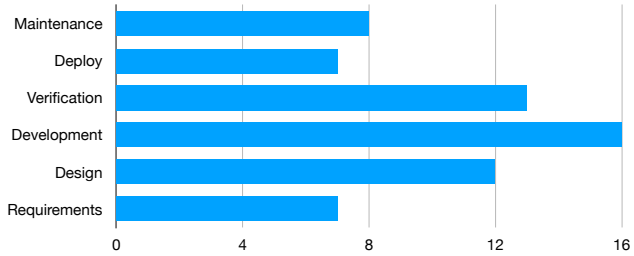


Fig. 2. Stages in which security testing is mainly done.

members. Among the 19 participants, 42% of them worked as developers or security specialists, 37% as testers, 10% as Project Managers, and the remaining had other roles—e.g., software architects and data scientists—currently. Moreover, we interviewed the CTO of one of the companies involved in the study, with many years of experience as a security engineer and manager. Furthermore, 84% of our sample was male, while 16% was female. Looking at these percentages, we could consider our sample realistic.[4] Additionally, this degree of gender diversity also allowed us to draw more diverse and generalizable conclusions. Last but not least, most of our participants have many years of experience in their role (59% affirm having at least ten years), and only a minority report having at least three years (21% of them). From these basic descriptive statistics, we can claim that the answers collected provide reliable insights for validating the information gathered from our interviews.

### B. $RQ_1$ – *Security Testing Arrangment*

According to our results, 12 participants said that security testing is performed by individuals or specific groups of people having the role of security testers, while seven participants declared that security testing is performed by the same developers that developed the software component but with knowledge and skills on security testing. Moreover, our interviews revealed a two-faced reality combining the answers to the second and third questions. First, seven participants reported that development teams do not adopt a rigorous approach to security testing procedures—for example, a framework like the MS-SDL [34] or OpenSAMM [8]. Nevertheless, teams adopting a systematic framework—e.g., the Microsoft Security Framework [21]—are characterized by several dedicated security officers performing the abovementioned activities. Furthermore, our semi-structured interview underlines how adopting a framework is not enough. Specifically, the participant said that "***Frameworks must be used by people who have the right skills; otherwise, developers may not consciously apply them and slow down the productivity of the entire team***". Indeed, such frameworks are complex instruments to master and adopt, and for this reason, selecting people with the right technical skills is crucial.

[4]Girls and women in STEM: https://www.industry.gov.au/news/second-national-data-report-on-girls-and-women-in-stem.

TABLE I
MOST USED SECURITY TESTING STRATEGIES.

| Rank | Phases | Strategy | Frequency in answers |
|---|---|---|---|
| 1 | Development | **Static Source Code Analysis.** The analysis of the application source code for finding vulnerabilities without actually executing the application. | 18 |
| 2 | Verification | **Automated Vulnerability Scanner.** An automatic technique with the aim of verifying if a specific component of the software has known vulnerabilities. | 16 |
| 3 | Design | **Architecture Security review.** A manual review of the product architecture to ensure it fulfils the necessary security requirements. | 14 |
| 4 | Requirements and Design | **Threat Modelling.** A structured manual analysis of an application-specific business case or usage scenario—guided by a set of precompiled security threats. | 13 |
| 5 | Verification | **Manual or Automated Penetration Testing.** A technique that simulates an attacker sending data to the application and observes its behaviour. | 13 |
| 6 | Development | **Static Binary Code Analysis.** The analysis of the compiled application (binary) for finding vulnerabilities without actually executing the application. | 10 |
| 7 | Verification | **Fuzz Testing.** A technique in which tools send random data, usually in larger chunks than expected by the application, to the input channels of an application to provoke its crashing. | 10 |

The semi-structured interview highlighted a third approach to security testing, which involves outsourcing the task to a specialized external actor. Specifically, some companies outsource all security testing activities to another company specialized in this aspect. Such a result matches the findings of other works focusing on security testing sub-activities, e.g., penetration testing and ethical hacking [2, 10].

### C. $RQ_2$ – *How security testing is performed*

First, we interviewed participants on the phases in which security testing is mostly performed. To have a common software development lifecycle for all participants, we previously asked them to describe the most general steps they experience during projects. Then, we generalized it into six steps, i.e., requirements, design, development, verification, deployment, and maintenance. This representation matches the classical and well-known *waterfall model* [6] adopted by all the three IT companies involved in this study. From their responses—summarized in Figure 2—we found that security testing is mostly performed during the development (16 responses), where the architecture and structure of the product are coded. In addition, we found that the security testing activity is also performed during the design and verification phases, which are activities directly related to the development phase: in these two phases, the developed tests are planned and executed. Furthermore, security testing is performed less during maintenance, deployment, and requirements.

Regarding the standards and frameworks applied for performing security testing, two arose during our interviews, i.e., the *Microsoft Security Development Life-cycle (MS-SDL)* and the *OpenSAMM* framework. Both frameworks are adopted to perform security testing, but the first is primarily preferred. The motivation for this can be found in the fact that a large and well-known company supports MS-SDL—i.e., Microsoft—that continuously updates the framework and performs improvements on it.

As for the strategies applied, we extracted the ones described in Table I, ordered by relevance—i.e., the ones with lower rank have been discussed and cited by most of the participants. The identified strategies perfectly match the most well-known procedures already identified by state of the art for mitigating and resolving security issues [4]. For this reason, our results contribute by (1) providing a qualitative confirmation of state-of-the-art findings and (2) ranking them by relevance. The most used strategies are (1) Static Source Code Analysis, (2) Automated Vulnerability Scanner, and (3) Architecture Security review—described in Table I. The previous question also confirms such ranking—the one whose results are described in Figure 2; indeed, the phases where these strategies are performed are also the phases in which security testing is mainly done—according to practitioners' opinion. The semi-structured interview confirmed that the techniques mentioned are widely used and provide guidelines to security experts to prevent security problems. In addition, it is important to emphasize that security experts and developers should together contribute to the creation of a well-defined development and security testing process. Static source code analysis tools represent the instruments that most easily allow security testers and developers to collaborate, as those tools are frequently included in continuous integration and testing pipelines: as such, the outcomes reported by static code analyzers can be visualized and manipulated by both the figures that, therefore, are more able to interact between each other and find quality assurance concerns, including security issues. Regarding the second and third strategies, they are specifically designed to identify security problems, and—for such a reason—it is normal that they are largely adopted.

As a final note, we were interested in exploring if security testing can be collaboratively managed as a team or whether, instead, it should be mostly considered as an individual activity performed by practitioners. With respect to this point, 13 interviewees reported that security testing could be a collaborative activity, especially when it revolves around the interaction between internal and external experts, e.g., in cases where the security testing processes are outsourced to external companies. In these cases, our semi-structured interview revealed that: *"**We need to engage companies experienced in security testing to verify our software. We used to include more than one third-party company to mitigate the learning effect that can be created over time**"*. However, given the high business value of software products and the need to optimize costs for third-party companies, the interviewee revealed the need to create an in-house ethical hacker team. The interviewee also highlighted that: *"**By doing so, developers have the support of guidelines to develop procedures. After the guidelines are defined, the internal ethical hacker team checks that everything is correct, and finally, we involve external security professionals**"*. For this reason, they need a *security manager* from the company side—or the development team—that communicates with ethical hackers and the third-party team to organize the activities. For this reason, the role of managers also changes, as the security manager will direct the activities between internal and external security teams and orchestrate communication with the project manager, who will lead the activities of the development team. Overall, communication between the development and security teams is managed by two managerial entities.

## V. Lessons Learned

Our study was conducted in the context of security testing to elicit managerial insights to improve this activity and increase the robustness of software products. On the basis of the findings collected in our study, we were able to distill a list of lessons learned which may be potentially useful to (1) practitioners, who may use our insights to make more informed decisions on the way security testing would be worth to be performed; and (2) researchers, who may want to take our findings as a base for building further knowledge on the matter.

First of all, our findings suggest that practitioners perceive security testing as an "elite" activity that must be performed by individuals with affirmed skills in both software engineering and security fields. Furthermore, only these individuals should use a security framework to ensure the consistency and correct execution of the activities. Hence, we can conclude that:

☑ *Only a specialized team of security experts should use security testing frameworks.*

Although most companies rely on third-party companies to perform security testing, managerial figures agree on the fact that assembling and training an internal team is the best way to ensure (i) a high-quality process and (ii) a continuous improvement tailored to the specific working context—which is essential for the quality improvement cycle [20].

☑ *Create an internal team that does security testing to enable quality improvement over time.*

Finally, security testing undoubtedly needs a managerial effort, especially in cases where a third-party company is involved. A dedicated manager should guide the security testing team and communicate with the development team manager, other than facilitating the coordination with third-party companies. In addition, the figures of security and project managers should be separated to ensure the correct and unbiased execution of the activity.

☑ *Arrange for two management figures, one for the development team and one for the security team, who communicate with each other.*

## VI. Threats to Validity

We report on them and discuss the possible confounding factors that might have influenced the results of our study.

### A. Threats to Construct Validity

Threats in this category are mainly due to imprecision in performed measurements [46]. In the context of our empirical study, these are mainly related to the way we designed the interviews. Starting from the objectives posed, we attempted to define clear and explicit questions in the survey that could allow participants to properly understand the meaning/phrasing and provide an answer that could have been directly mapped onto our objectives. The survey design involved the first three authors of the paper, who all have experience in software testing, empirical software engineering, and research design. While this already partially mitigated threats to construct validity, we also involved five external software engineering developers with work in the context of security testing. This additional investigation aimed to preliminarily assess our sample population's opinions, highlighting possible issues to be fixed before the large-scale survey was released. The pilot study let some minor concerns emerge that we promptly fixed. The follow-up double-checks of the involved developers let us be even more confident of the validity of our survey. Nonetheless, replications of the survey study would be beneficial to discover additional points of view and perspectives we might have missed. In this respect, we made all our data publicly available to make our results repeatable and reproducible [11].

### B. Threats to Conclusion Validity

Threats in this category are concerned with the ability to draw correct conclusions about relations between treatments and outcomes [46]. We conducted a content analysis to interpret the opinions of participants. In this respect, we systematically approached the matter with three researchers involved. In any case, we released all the material produced in the context of this study to enable the verifiability of the conclusions described in our online appendix [11].

Our findings are based on the perspective of the interviewees involved and must therefore be considered limited to the practitioners' own experience and opinions: while we argue that the insights of our work can already provide initial, significant reflections on the way security testing may be improved, we acknowledge the need for further studies aiming at analyzing the impact of the strategies indicated by our interviewees on multiple performance indicators, hence corroborating and extending the findings of our work. Part of our future research agenda plans for these extended investigations.

An additional point of discussion concerns the data filtering process conducted, which might have missed the exclusion of answers released by participants who did not have enough expertise to approach our study. In this respect, the paper's first author went through each response and validated it. Then, the second and third authors confirmed the operations performed. Such a double-check makes us confident of not considering answers that might have biased our conclusions.

### C. Threats to External Validity

Threats in this category are concerned with the generalizability of the results [46]. Our conclusions pertain to a sample of 19 participants (18 practitioners and one CTO). On the one hand, the amount of interviewees involved is our study is close to the one of other studies published in similar software engineering research contexts [9, 14]. On the other hand, our participants have the characteristics described in Section IV-A, i.e., they work in multinational IT security and development companies with at least 200 employees. Despite they were selected through the use of convenience sampling [16], the interviewees have a profile that characterizes them as experienced security experts working in a global software engineering context on the maintenance and evolution of large-scale software systems. As such, we are confident that our findings might be satisfactorily transferred to the more general population of security experts working under the same working conditions and context as those that took part of our study [15, 45]. Of course, replications of our work would corroborate our hypothesis—we provided an online appendix with all the relevant information to stimulate replications [11]. Part of our future research agenda aims at generalizing the findings obtained and verifying the extent to which the reported results might be actually transferred to a more general population of security experts.

The second main threat concerns the design of the interviews. In order to be sure to collect all the relevant information, we followed the guidelines provided by Andrews et al. [3], Hove and Ada [18], and Stol et al. [42] to define clear and explicit questions. Additionally, we conducted a pilot study with four developers that reported possible biases and flaws that we fixed before conducting the real interviews.

## VII. Conclusion

Our study aimed at enlarging the current knowledge of security testing in practice through a qualitative investigation. Despite being collected on a relatively low amount of practitioners, our results could already provide insights into the security testing practices adopted in the industry and the managerial aspects that might help address security.

To sum up, our work provides the following contribution:

1) We analyzed the security activities performed in business companies and how these strategies are integrated into the software life cycle;
2) We provided managerial insights on the figure of security tester in terms of technical skills;
3) We released a publicly available appendix [11] providing data and results of our study. Other researchers may use it to replicate and further extend our work.

Our future research agenda includes a larger-scale analysis of security testing in practice aiming at addressing potential threats to the conclusion and external validity of our work.

## Acknowledgments

This work has been partially supported by (i) the Swiss National Science Foundation - SNF Project No. PZ00P2_186090 (ii) the project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

## REFERENCES


[1] K. Abouelmehdi, A. Beni-Hessane, and H. Khaloufi, "Big healthcare data: preserving security and privacy," *Journal of big data*, vol. 5, no. 1, pp. 1–18, 2018.

[2] H. M. Z. Al Shebli and B. D. Beheshti, "A study on penetration testing process and tools," in *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. IEEE, 2018, pp. 1–7.

[3] D. Andrews, B. Nonnecke, and J. Preece, "Conducting research on the internet:: On- line survey design, development and implementation guidelines," 2007.

[4] R. Bachmann and A. D. Brucker, "Developing secure software: A holistic approach to security testing," *Datenschutz und Datensicherheit (DuD)*, vol. 38, pp. 257–261, 2014.

[5] S. Baltes and P. Ralph, "Sampling in software engineering research: A critical review and guidelines," *Empirical Software Engineering*, vol. 27, no. 4, p. 94, 2022.

[6] A. H. D. Bernd Brugge, *Object-Oriented Software Engineering Using UML, Patterns, and Java*, 2nd ed., 2004.

[7] S. Cavanagh, "Content analysis: concepts, methods and applications." *Nurse researcher*, vol. 4, no. 3, pp. 5–16, 1997.

[8] P. Chandra. Software assurance maturity model. [Online]. Available: https://opensamm.org/downloads/SAMM-1.0.pdf

[9] V. Chang, P. Baudier, H. Zhang, Q. Xu, J. Zhang, and M. Arami, "How blockchain can impact financial services–the overview, challenges and recommendations from expert interviewees," *Technological forecasting and social change*, vol. 158, p. 120166, 2020.

[10] M. Denis, C. Zena, and T. Hayajneh, "Penetration testing: Concepts, attack methods, and defense strategies," in *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. IEEE, 2016, pp. 1–6.

[11] D. Di Dario, V. Pontillo, S. Lambiase, F. Ferrucci, and F. Palomba, "Security testing in the wild: An interview study – online appendix," 2023. [Online]. Available: https://drive.google.com/drive/folders/1LqHwc5vA8lmrLAlJueO750TSNYOX2f8b?usp=share_link

[12] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, "Software security patch management-a systematic literature review of challenges, approaches, tools and practices," *Information and Software Technology*, vol. 144, p. 106771, 2022.

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

[14] H. Frluckaj, L. Dabbish, D. G. Widder, H. S. Qiu, and J. HERBSLEB, "Gender and participation in open source software development," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, pp. 1–31, 2022.

[15] S. Ghaisas, P. Rose, M. Daneva, K. Sikkel, and R. J. Wieringa, "Generalizing by similarity: Lessons learnt from industrial case studies," in *2013 1st International Workshop on Conducting Empirical Studies in Industry (CESI)*. IEEE, 2013, pp. 37–42.

[16] J. F. Hair, A. H. Money, P. Samouel, and M. Page, "Research methods for business," *Education+ Training*, vol. 49, no. 4, pp. 336–337, 2007.

[17] S. Hosseinzadeh, S. Rauti, S. Laurén, J.-M. Mäkelä, J. Holvitie, S. Hyrynsalmi, and V. Leppänen, "Diversification and obfuscation techniques for software security: A systematic literature review," *Information and Software Technology*, vol. 104, pp. 72–93, 2018.

[18] S. Hove and B. Anda, "Experiences from conducting semi-structured interviews in empirical software engineering research," in *11th IEEE International Software Metrics Symposium (METRICS'05)*, 2005, pp. 10 pp.–23.

[19] M. Howard and S. Lipner, *The security development lifecycle*. Microsoft Press Redmond, 2006, vol. 8.

[20] P. M. Institute, *A Guide to the Project Management Body of Knowledge*, 7th ed., 8 2021.

[21] J. Jones. (2008) Windows vista one year vulnerability report. [Online]. Available: https://www.microsoft.com/security/blog/2008/01/23/download-windows-vista-one-year-vulnerability-report

[22] R. A. Khan, S. U. Khan, M. Ilyas, and M. Y. Idris, "The state of the art on secure software engineering: A systematic mapping study," *Proceedings of the Evaluation and Assessment in Software Engineering*, pp. 487–492, 2020.

[23] J. C. Knight, "Safety critical systems: challenges and directions," in *Proceedings of the 24th international conference on software engineering*, 2002, pp. 547–550.

[24] M. N. Kreeger, "Security testing: Mind the knowledge gap," *ACM SIGCSE Bulletin*, vol. 41, no. 2, pp. 99–102, 2009.

[25] M. Kreitz, "Security by design in software engineering," *ACM SIGSOFT Software Engineering Notes*, vol. 44, no. 3, pp. 23–23, 2019.

[26] I. Krsul, "Software vulnerability analysis," *ETD Collection for Purdue University*, 01 2011.

[27] I. V. Krsul, *Software vulnerability analysis*. Purdue University, 1998.

[28] M. Kumar and A. Sharma, "An integrated framework for software vulnerability detection, analysis and mitigation: an autonomic system," *Sādhanā*, vol. 42, pp. 1481–1493, 2017.

[29] P. Kumar and H.-J. Lee, "Security issues in healthcare applications using wireless medical sensor networks: A survey," *sensors*, vol. 12, no. 1, pp. 55–91, 2011.

[30] D. R. McKinnel, T. Dargahi, A. Dehghantanha, and K.-K. R. Choo, "A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment," *Computers & Electrical Engineering*, vol. 75, pp. 175–188, 2019.

[31] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, 3rd ed. Wiley Publishing, 2011.

[32] M. Pezzè and M. Young, *Software testing and analysis: process, principles, and techniques*. John Wiley & Sons, 2008.

[33] B. Potter and G. McGraw, "Software security testing," *IEEE Security & Privacy*, vol. 2, no. 5, pp. 81–85, 2004.

[34] B. Potter, "Microsoft sdl threat modelling tool," *Network Security*, vol. 2009, no. 1, pp. 15–18, 2009.

[35] B. Potter and G. McGraw, "Software security testing," *IEEE Security & Privacy*, vol. 2, no. 5, pp. 81–85, 2004.

[36] S. Rafique, M. Humayun, Z. Gul, A. Abbas, H. Javed *et al.*, "Systematic review of web application security vulnerabilities detection methods," *Journal of Computer and Communications*, vol. 3, no. 09, p. 28, 2015.

[37] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting devsecops: A systematic review," *Information and software technology*, vol. 141, p. 106700, 2022.

[38] M. C. Sánchez, J. M. C. de Gea, J. L. Fernández-Alemán, J. Garcerán, and A. Toval, "Software vulnerabilities overview: A descriptive study," *Tsinghua Science and Technology*, vol. 25, no. 2, pp. 270–280, 2019.

[39] M. Sánchez-Gordón and R. Colomo-Palacios, "Security as culture: a systematic literature review of devsecops," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 266–269.

[40] I. Schieferdecker, J. Grossmann, and M. Schneider, "Model-based security testing," *arXiv preprint arXiv:1202.6118*, 2012.

[41] D. Stahl, T. Martensson, and J. Bosch, "Continuous practices and devops: beyond the buzz, what does it all mean?" in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2017, pp. 440–448.

[42] K.-J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research: A critical review and guidelines," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 120–131. [Online]. Available: https://doi.org/10.1145/2884781.2884833

[43] H. Thompson, "Why security testing is hard," *IEEE Security & Privacy*, vol. 1, no. 4, pp. 83–86, 2003.

[44] K. Tuma, G. Calikli, and R. Scandariato, "Threat analysis of software systems: A systematic literature review," *Journal of Systems and Software*, vol. 144, pp. 275–294, 2018.

[45] R. Wieringa and M. Daneva, "Six strategies for generalizing software engineering theories," *Science of computer programming*, vol. 101, pp. 136–152, 2015.

[46] C. Wohlin, M. Höst, and K. Henningsson, "Empirical research methods in software engineering," in *Empirical methods and studies in software engineering*. Springer, 2003, pp. 7–23.

[47] C. Wysopal, L. Nelson, D. D. Zovi, and E. Dustin, *The Art of Software Security Testing: Identifying Software Security Flaws (Symantec Press)*. Addison-Wesley Professional, 2006.

[48] J.-P. A. Yaacoub, H. N. Noura, O. Salman, and A. Chehab, "Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommendations," *International Journal of Information Security*, pp. 1–44, 2022.