UNIVERSITÀ DEGLI STUDI DI SALERNO

Computer Science Department

Master's Degree in Computer Science

GRADUATION THESIS

Security Testing in the Wild: An Empirical Study into the Security Testing Methodologies in Practice

SUPERVISOR

Prof. **Fabio Palomba**

Dr. **Emanuele Iannone**

Dr. **Stefano Lambiase**

Dr. **Valeria Pontillo**

CANDIDATE

Dario Di Dario

Matricola: 0522500848

Academic year 2021-2022

I nonni ti vedono crescere, sapendo che ti lasceranno prima degli altri.

Forse é per questo che ti amano più di tutti...

A nonno Manfredi, nonna Teresa e nonno Aldo.

Abstract

Today's information systems are increasingly complex and full of different technologies. As a result, they can run into security problems if they are not developed with proper organization. One of the most important security threats is source code vulnerabilities, which create damage, sometimes catastrophic, to the software and the company that owns it. One example is that of the company *CardSystem*, responsible for processing Visa, MasterCard, and Maestro credit cards. However, in 2005 it was discovered that data from over 40 million credit cards had been leaked due to an internal security problem. One of the reasons was the lack of a security testing expert. Therefore, we performed an empirical study to determine whether current companies have introduced security testing experts with the increase in technologies. More in detail, we want to determine whether the same companies use the current security frameworks described in the literature and how they fit in their contexts. The survey has been sent to three different companies, which differ in size and development contexts—e.g., human resource software, automotive software, and accountant software. In the end, an individual interview with one of the three companies was conducted to obtain relevant details. The results highlight that two of the three companies incorporate the figure of the security tester and that the entire process—albeit through modifications in relation to the company structure—utilizes frameworks based on the security lifecycle. From a management perspective, we found that companies that implement managerial figures in communication between development and security tester teams perform better than others.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation and Context

Modern IT software has to operate in different contexts and for different purposes and, for such a reason, is subject to security attacks that can cause high severity damage. In order to improve the maintainability and error prevention of such complex systems, state of art defined the concept of *vulnerability*. A vulnerability is an **instance of fault** in the specification, development, or configuration of software such that its execution can violate the (implicit or explicit) security policy defined within a software organization [1]. There is an increasing awareness of software security needs, and the demand for security systems is increasing. A company can lose revenue or eventually go out of business if its products are considered insecure. In June 2005, more than 40 million debit and credit cards were exposed to criminals when *CardSystems* was a victim of a cyber break-in. Forensic analysis after the theft revealed that CardSystems was not handling sensitive data properly. Its software was making extra copies of the credit card data and storing it unencrypted, making it easy pickings for cybercriminals. This prompted big customers to reevaluate their relationship with CardSystems, and four months later, the company was sold. The main reason for this disaster is that the company did not have experienced security testers assigned to

carry out security activities, following a proper organizational process. They were also convinced that security problems could be remedied through software testing activities. So, there seems to be a misconception about who should perform security testing and who should perform software testing [2]. Software Testing and Security Testing cannot be defined as the same thing because of the different points of view: while quality assurance (QA) tests the system from the perspective of a standard or slightly curious user, security tests it from the perspective of a malicious user. To give a more concrete example, let us consider how each would test a standard login form: QA Testing might conduct the test in order to verify what would happen if the user accidentally mistypes their email address while logging in. On the other hand, a security tester might conduct the test to verify that the form is not vulnerable to SQL Injection, i.e., a common security issue that can have devastating effects when found on login forms. They test input validation and error handling on the login screen in both cases, but their intentions differ. Since QA tests normal behavior and security testers test malicious behavior, sometimes the words "use case" and "misuse/abuse case" are used respectively to refer to their test cases.

For this reason, security should be addressed from the beginning—already during system inception—with defined security policies and use cases by security testers. In addition, security should be built into a program during the design stage and coded by developers trained in secure programming. Nowadays, different frameworks allow companies to follow detailed guidelines on how to perform security testing, such as **Microsfot Secure Development Life-cycle** and **OpenSAMM from OWASP**. This thesis work aims to provide both academic and professional contributions. We carefully evaluated the literature on security testing methodologies and techniques needed within the security life cycle. We conducted a survey with three IT companies working in different contexts. Results show that security testing practices are applied and how the entire process is carried out.

1.2 Goal and Results

For the reasons mentioned above, we conducted a qualitative investigation to deepen the application of security testing standards in practitioners' fields. From

a practical point of view, we used both a survey and an interview method: first, we interviewed 26 specialists between developers and managers to gather general knowledge about the topic; then, we strengthened our results and augmented them with more focused and personal investigations.

The results of our study reveal that most of the companies surveyed use security frameworks, but it also depends on the companies' internal structure and size. What varies are the types of techniques to be used depending on the applications to be developed. Most companies believe that testing activities should be entrusted to people with backgrounds in security and that both the development team and the security testing team should collaborate throughout the project life cycle. These results indicate that security testing is still far from being a standardized practice for every company. Such a situation leads to problematic conditions—e.g., dissemination of strictly personal data—with potentially catastrophic results. For such a reason, project managers should consider security testing as a first-class citizen not only when shaping their own offshoring activities but the same aspects need to become prime during all the project life-cycle phases. In addition, further research is needed to create additional guidelines for making a single standard. In addition, it would be helpful to develop a source code analysis tool to demonstrate the prediction of dangerous code. The tool will be able to support the tester in using the proper security testing techniques.

1.3 Thesis Structure

This Master thesis could be structured as follow:

- **Chapter 2 - Background**: A general vision of what macro-area this work could be used, with an high level panoramic of **Software Engineering** and **Software Dependability** aspects, as well as **Software Testing** and **Security Testing**.
- **Chapter 3 - State of the Art**: It provides relevant details on the state of practice regarding security testing as well as Software Engineering practices. In fact, some theoretical frameworks created to enable users to approach secure software development, through its life cycle, are described. Examples and

methodologies of security testing for each phase of the software life cycle will also be presented.

- **Chapter 4 - Design**: Design and Development chapter provide details about the research goals and the empirical investigation, as well as aspects related to the survey design and its submissions to different group of company workers.
- **Chapter 5 - Analysis of the Results**: This chapter wants to describe the results analysis process, more in specific our work is to collect all the procedures useful for the processing of the data obtained from the survey, with respect to data-cleaning and pre-processing methodologies and the original data and finally, move on to the real analysis of the and generalization of the results with related points of discussion and implication.
- **Chapter 6 - Threat To Validity**: This chapter describes and illustrates the threats to the validity of the study and the way we have mitigated them.
- **Chapter 7 - Conclusions and Future Work**: Finally, this chapter takes up the findings of the previous chapter to understand whether state-of-the-art methodologies are used in the practices of different companies and how the methodologies are modified depending on the type of software the company is used to develop.

CHAPTER 2

Background

Technological advances and the complexity of using software are increasing daily. The same technological progress is also intended to use and use complex development models that allow us to develop high-quality software. One aspect that is often underestimated is software security. Software security testing is a software testing process that ensures the software is free of any potential vulnerabilities or weaknesses, risks, or threats so that the software might not harm the user system and data. For this reason, performing software security tests, often multiple times, is essentially a prerequisite for publishing software today.

2.1 Software Engineering

Software engineering is an engineering discipline concerned with all aspects of software production, from the early stages of system specification to maintaining the system after it has been used. In this definition, there are two key phrases:

1. *Engineering discipline* Engineers make things work. They apply theories, methods, and tools where appropriate, but they use them selectively and always try to discover solutions to problems even when there are no applicable theories

and methods. Engineers also recognize that they must work with organizational and financial constraints, so they look for solutions within these constraints.

2. *All aspects of software production* Software engineering is concerned not only with the technical processes of software development but also with activities such as software project management and the development of tools, methods, and theories to support software production.

Software engineers generally adopt a systematic and organized approach to their work, which is often the most effective way to produce high-quality software. However, engineering is all about selecting the most appropriate method for a set of circumstances and a more creative, less formal approach to development may be effective in some cases. Less formal development is particularly suitable for developing web-based systems, which requires a blend of software and graphical design skills [3].

Bernd Brugge and Allen H. Dutoit in [4] take into account the different perspectives of software engineering, which are:

- *Modeling*: The purpose is to describe and understand complex systems focusing only on relevant details without any other aspect by using different kinds of **abstraction**[1] strategy [5].

- *Problem Solving*: Engineers search for an appropriate solution, often by trial and error, evaluating activities empirically, with limited resources and incomplete knowledge.

- *Knowledge Acquisition*: A common mistake that software engineers and managers make is to assume that the knowledge acquisition needed to develop a system is linear. Instead, knowledge acquisition is a non-linear process which means that adding new pieces of information may invalidate all the knowledge acquired for the system comprehension.

[1] Abstraction level: Conceptual, Requirements, Design, Implementation. These are used in the re-engineering concept to create models of software development as a sequence of phases, where the map of the steps to specific levels of abstraction and refinement

- *Rationale*: This is a non-trivial step because developers make any decision based on their experience and their intuition without explicitly evaluating different alternatives. However, software engineers must address the challenge of capturing and accessing rationale to deal with changing systems.

Every software product can be characterized by its *Software Development Life-Cycle* (SDLC) as well as *Maintenance and Evolution process*. The SDLC is a process used by the software industry to design, develop and test high-quality software. The SDLC aims to produce high-quality software that meets or exceeds customer expectations and reaches completion within times and cost estimates. The ISO/IEC 12207:2007 is the international standard for software life-cycle processes: it contains processes, activities, and tasks applicable during the acquisition, supply, development, operation, maintenance, or disposal of software systems, products, and services [6]. These processes are realized by stakeholder engagement to reach customer satisfaction.

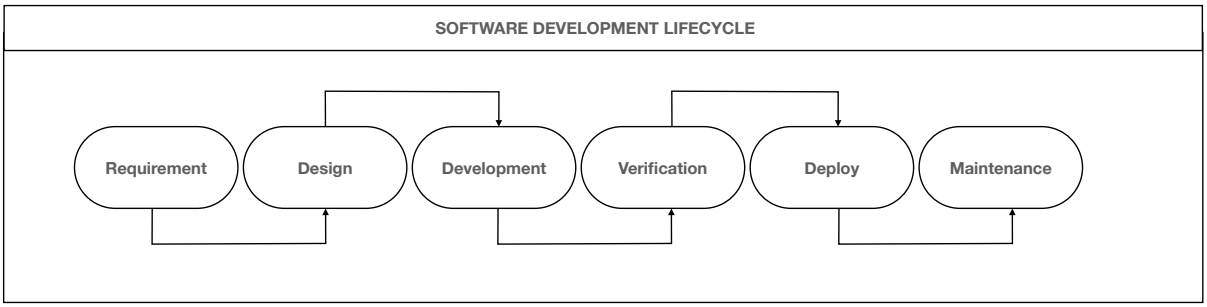

Figure 2.1: Software Development Life-Cycle

Figure 2.1 shows the phases for the SDLC, which are:

- *Requirement*: A requirement describes with non-formal language what the system should do. Instead, the process of finding, analyzing, documenting, and checking these services and constraints is called **Requirements Engineering** (RE).

- *Design*: This phase models how a software application will work, including design aspects as: **Architecture**, **User Interface**, **Platform**, **Programming method for solving a problem** and so on.

- *Development*: This is the actual writing of the program. A small project might

be written by a single developer, while a large project might be broken up and worked on by several teams.

- *Verification & Validation*: Testing is an activity in which a system or a system component can be executed under certain conditions, intending to observe and evaluate the results.

- *Deploy*: The deployment makes an application available to final users, but it can also be complex. For example, if we want to upgrade a company-wide database to a newly-development application because could be the presence of different other systems that use the database.

- *Maintenance*: At this point, the development is almost finished, but the maintenance phase takes up to 70% of the entire work. It is required to reinforce the system following the **Lehman's law** [7]. Different maintenance classification activities resolve different aspects (i.e., corrective, adaptive, perfective, preventive).

Thus, the software process model (*aka process paradigm* can be used based on SDLC. Ian Sommerville [3] defines a software process model as a simplified representation of a software process. Each software model represents a process from a particular perspective and, for this reason, it provides only some information about the process. Ruparelia et al. [8] say that it is important to keep in mind the difference between **model** and **methodology**, in which the former – as described above – talks about "what to do", while the latter talks about "how to do". Results show that every SDLC model assumes a specific relevance respecting the particular type of software project to be developed. Although the literature is full of SDLC models, for this study, we want to consider the most used by practitioners and researchers' points of view. These are: **Waterfall**, **spiral**, **unified**, **incrementing**, **rapid application development**, **the V** and **the W model**. However, the combination between *iterative* and *incremental* process bring the **Agile SDLC**[2] to life, that focuses on process adaptability and customer satisfaction by rapid delivery of working software product [9].

[2]More information: https://www.agilealliance.org

2.2 Software Testing

Let's focus on the fourth step of the software development life-cycle: **Verification & Validation** (V&V). It was introduced in 1990 to increase software quality and stakeholder reliability. While the requirements phase wants to specify the main users' possible operations with the software, the testing phase intends to test and prove the intent by discovering potential issues or defects in the code before the start of the deployment phase. The distinction between Verification and Validation was eloquently explained by the pioneer of software engineering Barry Boehm in 1979 [10], with the following questions:

- **Validation**:*"Am I building the right product?"*
- **Verification**:*"Am I building the product right?"*

Verifying that the software built conforms to its specifications and provides the functionality expected by those who pay for the product is the goal of verification and validation processes. Usually, the first question must be answered when a software engineer has to "translate" non-formal requirements into formal requirements and, for this reason, may request feedback. As soon as requirements are made public, these verification procedures begin and continue throughout the development process. The second question of Boehms [10] must be answered through the test, manual inspection, and static analysis without involving the customer. Checking that the software satisfies its declared functional and non-functional requirements is the goal of the verification phase.

2.2.1 Verification Phase: Static VS Dynamic

The verification and validation process can include software inspections and reviews in addition to software testing, such as system requirements, design patterns, program source code, and even proposed system tests. This type of activity is also called "static" V&V technique because it does not require software execution to be verified. Figure 2.2 depicts how software inspections and testing aid V&V at various stages of the software development process. The arrows point to the stages

of the process where the techniques can be applied. Although most inspections focus on a system's source code, any readable representation of the software, such as its requirements or a design model, can be inspected. To detect errors in a system, you use knowledge of the system, its application domain, and programming or modeling language. Based on [3] Software inspection has three advantages over testing:

1. Errors can mask (hide) other errors during testing. When an error causes unexpected outputs, you never understand whether the subsequent output anomalies result from a new error or are a side effect of the original error. There is no need to worry about relationships between errors when a developer uses the inspection code because it is a static procedure. For this reason, numerous problems in a system can be found during a single inspection session.

2. Incomplete versions of a system can be inspected at no extra charge. If a program is incomplete, you must create specialized test harnesses to test the available parts. This raises the costs of system development.

3. In addition to looking for program defects, an inspection can look at a program's overall quality attributes, such as standard compliance, portability, and maintainability. Inefficiencies, ineffective algorithms, and a poor programming style make the system difficult to maintain and update.

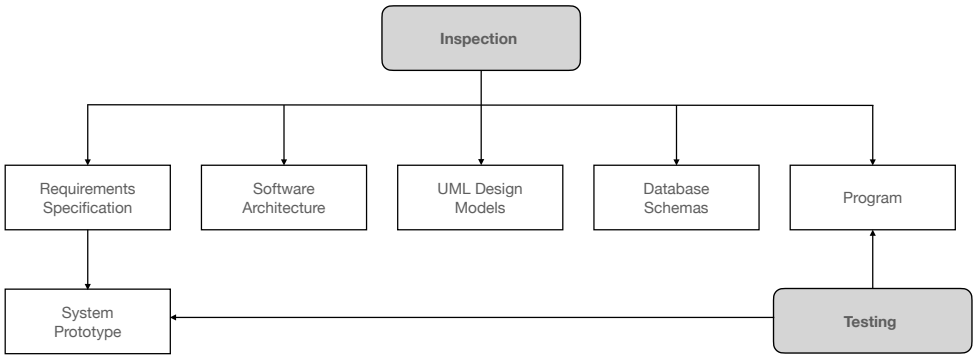

Figure 2.2: Inspection and Testing

Mike Fagan in [11] has reported that more than 60% of errors in a program can be detected using informal program inspection. Inspections, however, cannot take the role of software testing because they are ineffective in detecting errors caused

by unexpected interactions between program components, timing issues, or system performance issues. Furthermore, creating a separate inspection team can be difficult and costly, especially in small development groups, as all potential team members may also be software engineers. Nowadays, software testing plays an important role in software engineering and SDLC. It is referred to as the "dynamic" V&V technique because they require the software to be executed to be verified. All were born in 1951 by Joseph Juran, who is now considered to be the father of software testing. For the first time marked the importance of software quality assurance in his book "*Quality Control handbook*" [12], in 1990, there was the transition from software testing to a more comprehensive process called **Quality assurance** (QA), which covers the entire software development life-cycle and affects the processes of planning, design, creation, and execution of test cases, support for existing test cases and test environments. Myers in [13] has noticed that one of the key causes of bad program testing is that most programmers start with a misunderstanding of the phrase. They could say:

1. *"Testing is the process of demonstrating that no errors exist."*
2. *"The goal of testing is to demonstrate that a software does its intended functions accurately."*
3. *"Testing is the process of generating assurance that a software will perform as expected."*

These definitions are flipped. When testing a program, you want to add value to it. Adding value through testing involves improving the program's quality or reliability. Finding and correcting faults is part of increasing the program's reliability.

Definition

Software Testing is the process of executing a program with the intent of finding errors.

But developers, testers, project managers, and all other team components, tend to be highly goal-oriented and this can represent damage in the context of software testing because if the goal is to demonstrate that software has no error, we are conducted to select test cases with test data without any error. This argues that

testing is a destructive, even sadistic, process, which explains why most people reject it. Another strategy to reinforce the correct definition of testing is to examine how project managers use the words "successful" and "unsuccessful" in categorizing test case results. A test case that did not identify an error is usually referred to as a "successful test run," whereas a test that discovered a new error is usually referred to as "unsuccessful." This is, once again, flipped. "Unsuccessful" refers to something painful or disappointing. In Myers' opinion, a well-designed and executed software test is successful when it reveals problems that can be corrected. That same test is also successful when it determines that no more errors will be discovered. The only unsuccessful test fails to investigate the software adequately, and in most circumstances, a test that identified no errors would be considered unsuccessful because the concept of **a program without errors is essentially impractical**. For this reason, developers and testers cannot answer the question *"Did we find all the bugs?"*. E. W. Dijkstra once said:

"Testing can prove the presence of a bug, but not their absence"

There can't be an algorithm that proves the detection of all bugs, and you can't count the number of bugs that aren't present. This implies that testing itself is **undecidable**. Considering the undecidable of testing, the main purpose must be (i) to maximize failure identification; (ii) to minimize test case number. Testing techniques are necessary depending on the type of problem:

1. **Systematic Testing**: building test to discover defects inside the system. The most important two technique are (i) Functional Testing (*Black-Box*); (ii) Structural Testing (*White-Box*).

2. **Statistical Testing**: a test that reflects the user's frequency input and should be used for reliability estimation.

3. **Mutational Analysis**: Allows evaluating the goodness of testing.

4. **Random Testing**: Every input taken randomly from the user has the same probability of discovering faults in the program.

2.2.2 Construction Testing

We now discuss the testing techniques related to object-oriented programming. As discussed in Section 2.1, one of the most used SDLC models in software testing is the **V-Model**. Figure 2.3 shows two different phase:

- The **left-phase** of the model is Software Development Life-Cycle.
- The **right-phase** of the model represents the so-called Software Testing Life-Cycle (STLC).

This is a cost-effective strategy because the programmer is familiar with the component and is the appropriate person to produce test cases. If an incremental development strategy is adopted, each increment should be tested as it is built, with these tests based on the requirements for that increment. Tests are created with the requirements in extreme programming before development begins. These test plans generated from the system specification and design, are used by an independent team of testers. At this point, our focus goes to the **right-phase** of the model.

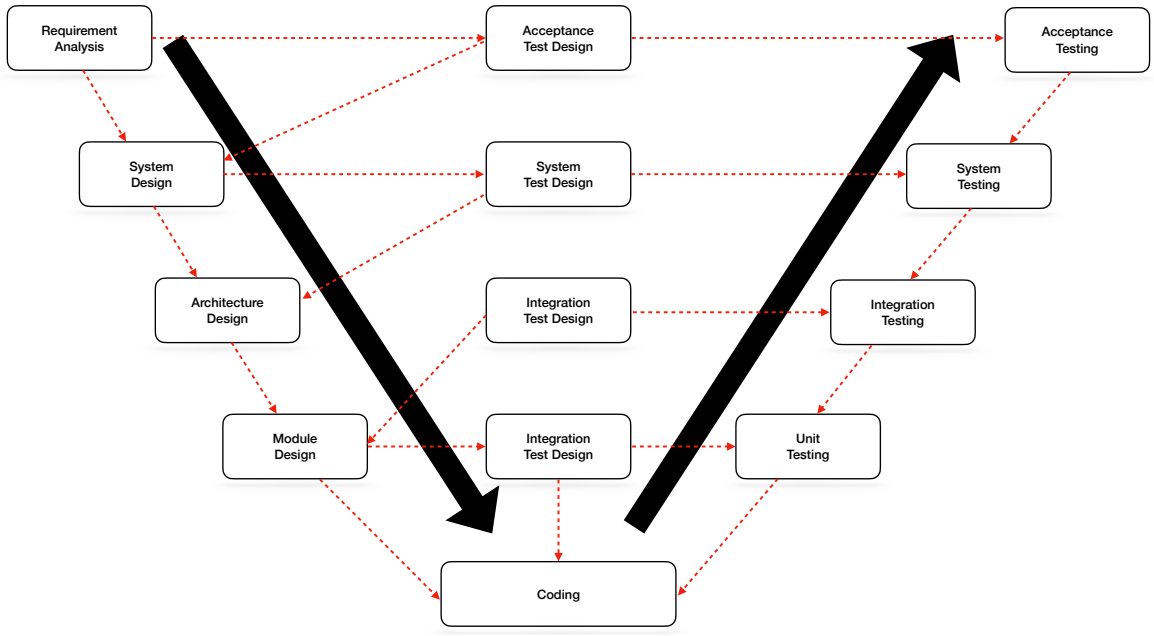

Figure 2.3: V-Model representation

2.2.3 Unit Testing

Unit tests —in the object-oriented domain — test the functionality of separately and in isolation testable pieces of software. Depending on the situation, it can be individual subprograms or a larger component consisting of highly cohesive components. Unit tests are generally performed with access to the code to be tested and with the assistance of debugging tools. Unit tests are often, but not always, performed by the programmers who wrote the code [14].

2.2.4 Integration Testing

The practice of evaluating the interactions of software components is known as integration testing. Top-down and bottom-up integration testing methodologies are frequently used with hierarchically structured software. Modern, architecture-driven integration solutions often entail iteratively integrating software components or subsystems based on identified functional threads. Integration testing is a common continuing practice at each step of development in which software developers abstract away lower-level perspectives to focus on the perspectives of the level at which they are integrating. Except for tiny, simple software, gradual integration testing methodologies are usually preferable over putting all components together simultaneously, sometimes known as *Big Bang* testing [14].

2.2.5 System Testing

System testing is concerned with testing the overall behavior of a system and it is commonly used to evaluate the system's behavior. Many software flaws will have been found through effective unit and integration testing. External interfaces to other applications, utilities, hardware devices, or operating systems are typically reviewed at this level as well [14].

2.3 Security Testing

Security testing is often not performed at the right time. This means that security and software testing activities are often performed later in the software development life cycle (SDLC). This increases the likelihood of adding vulnerabilities in the code. Vulnerabilities are software properties that occur during the design and implementation stages. According to Krsul [15] "*A vulnerability is an* ***instance of fault*** *in the specification, development, or configuration of software such that its execution can violate the (implicit or explicit) security policy defined within a software organization.*". It **is a weakness** which allows an attacker to reduce a system's information assurance. By doing so, attackers might *exploit* and take advantage of something for one's end, especially unethically or unjustifiably.

Some vocabulary on the life-cycle of vulnerabilities:

1. **Discovery**: When a vulnerability is discovered by a vendor, a hacker, or any third-party software analyst.

2. **Disclosure**: A vulnerability is publicly disclosed, i.e., everyone is informed about it.

3. **Exploitation**: An external attacker can use the vulnerability to cause issues.

4. **Patching**: When a vendor fixes the vulnerability.

In order to make developers increasingly aware of the presence of vulnerabilities within the code, the following was born it was created *"The Open Web Application Security Project"* (OWASP),[3] this non-profit organization aims to improve the safety of software. The OWASP Foundation serves as a resource for developers and engineers to secure the web through community-led open-source software projects, hundreds of local chapters globally, tens of thousands of members, and premier educational and training conferences.

Every year, the OWASP Foundation produce the **OWASP Top 10** which is a standard awareness document for developer and web application security. This list represents the most critical security risks to the web application. For example in

[3] `https://owasp.site-ym.com`

2020 **injections**, **broke authentication**, and **cross-site scripting** vulnerabilities are included in the list. But the most interesting thing concerns the *"Using components with known vulnerabilities"* problem. Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. In this manner, an attacker can facilitate serious data loss or server takeover.

Definition

Security Testing is testing of security requirements related to security properties like confidentiality, integrity, availability, authentication, and authorization.

During software development, it may happen to encounter bugs in the code. In the context of object-oriented programming, for example, a bug report may identify one or more classes with problems within them. In the context of vulnerabilities, it is useful to perform an analysis in previous versions of the source code with the aim of highlighting when the error was inserted. The same idea should be carried out using the secure development life-cycle.

Thus, as soon as the project starts, secure software development must begin. The security testing stage serves as the last "security gate" for an application in many software-development organizations, authorizing or discouraging the transition from the relaxed setting of software engineering to the real, "unsafe," world. A significant responsibility comes with this phase, which occurs late in the software development life-cycle: Application security and the company's reputation may depend on it [2].

2.3.1 Secure Software Development Life-cycle

One method that can attenuate the security cost correction is to use a **Secure Software Development Life-cycle** (SSDL). The SSDL strictly follows the SDLC structure and concept it is important to compare Figure 2.4 and Figure 2.1.

Wysopal et al. in [2] say that SSDL has six primary components, which are:

- **Phase 1**: Security Guidelines/Rules and Regulation
- **Phase 2**: Security Requirements
- **Phase 3**: Architectural Reviews/Threat Modeling

Table 2.1: Cost to correct error

Phase	Relative cost to correct
Definition	$1
High-Level Design	$2
Low-Level Design	$5
Code	$10
Unit Testing	$15
Integration Testing	$22
System Testing	$50
Post-Delivery	$100

- **Phase 4**: Secure Coding Guidelines
- **Phase 5**: White/Gray/Black Box Testing
- **Phase 6**: Determine Exploitability

Phase 1: Security Guidelines/Rules and Regulation

Security standards, laws, and ordinances must be taken into account when planning a project. The SSDL's initial phase is known as the "umbrella requirement." The security standards that apply to the system are defined by a system-wide specification, which may be based on certain governmental legislation. For example, the ones were used by Audi Group [4] to define the rules for information security that system operators and administrators must follow when handling information and IT devices. In addition, the Information Security Guideline for employees or third parties provided that the system operator or administrator is an employee of a partner company applies to the target group of system operators and administrators. Moreover, the documents follow the international standard ISO/IEC 27002:2013.

The Web Application Security Standards (WASS) initiative intends to develop a proposed list of minimal standards a Web application must follow if it processes

[4] `https://bit.ly/3QJWFn3`

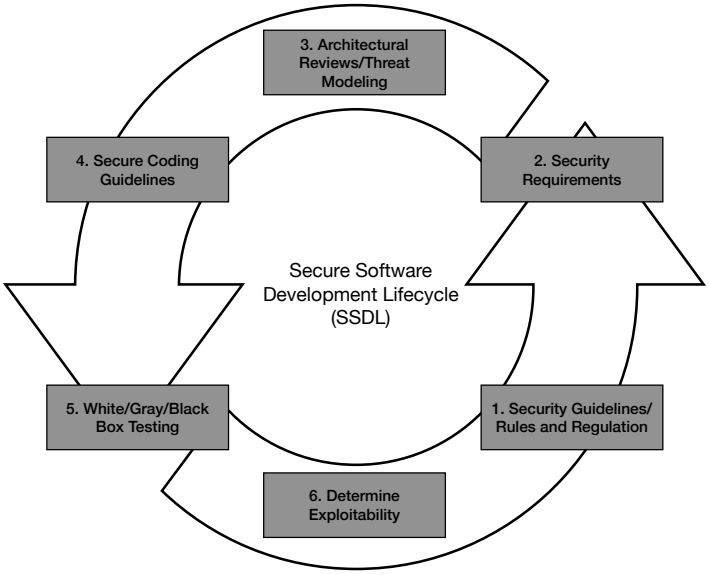

Figure 2.4: Secure Software Development Life-cycle

credit card information, and it is also recommended by OWASP. The objective of this project is to create precise, verifiable criteria that may either be used independently or in conjunction with other security standards already in place. So, it should be able to tell if a Web-based application has been developed using best practices and minimal security measures by testing it against this standard. Some systems are free from all applicable laws and regulations. A security policy still needs to be created in those circumstances. The security policy should be documented, but it should also be followed and continually assessed to be enforced.

Phase 2: Security Requirements

The second level of the SSDL is security requirements. Omitting security requirements from any type of requirements documentation is a typical mistake, but it is critical to record security requirements. Not only do security requirements help in the design, implementation, and creation of software test cases, but they can also help in choosing technologies and identifying areas of potential risk. The security engineer must insist that each functional requirement include a description and documentation of the corresponding security requirements. A section titled "Security Requirements" should be included in each functional requirement description to record any specific security requirements that differ from the general security policy

or specifications for the system. To ensure that the system is designed appropriately, rigorous analysis is required for all but the smallest programs. One method of describing security requirements is the use of attack cases. Use cases can lead to more accurate test protocols and system designs. Everyone would agree, for example, that "The system must be very secure," but everyone would have a different definition of what that means. Security requirements do not provide the system with any special capabilities. Instead, they impose limitations or go further, specifying how the system will deal with any function that should not be allowed. Security defect prevention is the use of techniques and processes that can help detect and avoid security errors before they propagate to later stages of development. If safety is considered throughout the development life-cycle, everyone will be better able to identify omissions, discrepancies, ambiguities, and other problems that could compromise the safety of the project. Now let's give some (nonformal) security requirements examples:

- The application transmits sensitive user information across potentially untrusted or unsecured networks. To protect the data, communication channels must be encrypted to prevent snooping, and mutual cryptographic authentication must be employed to prevent man-in-the-middle attacks.

- The application supports multiple users with different levels of privilege. The application assigns users to multiple privilege levels and defines the actions each privilege level is authorized to perform. The various privilege levels need to be defined and tested. Mitigations for authorization bypass attacks need to be defined.

- The application takes user input and uses SQL. SQL injection mitigations are a requirement.

- User input must be validated for length and characters.

Phase 3: Architectural Reviews/Threat Modeling

The third stage of the SSDL consists of reviews of architectural and design elements as well as threat modeling. To create stronger and more comprehensive security strategies, plans, designs, procedures, and methodologies, security practitioners need

a clear understanding of the product's architecture and design. Early security team involvement can assist avoid weak security designs and insecure architectures, as well as later project life-cycle misunderstanding about the application's behavior. Additionally, early involvement enables the security specialist to understand which components of the application are the most crucial and which pose the greatest security risks. This information enables security professionals to concentrate on the application's most crucial components first and assists testers in avoiding over-testing low-risk areas and under-testing high-risk ones. Threat modeling has advantages over code reviews and testing in that it can identify higher-level design problems as opposed to implementation bugs. Before they are coded into products, security issues might be discovered early on here. This aids in identifying the application components that pose the greatest risk and require the most attention during the software development process.

Phase 4: Secure Coding Guidelines

Implementation vulnerabilities are carried on by security flaws in the software's actual coding. By inspecting the source code or the binary executable, static analysis tools can find several implementation issues. These tools are very helpful in identifying problems as buffer overflows, and the information they produce can teach programmers how to avoid errors entirely. It is advised that testers and software developers take part in training programs on how to write secure code by following these secure coding guidelines. Testers can then create test cases to ensure that the standard is being met by creating baselines using secure coding standards. Additionally, there are sites where you may send your code to have a neutral third party check it for errors. Utilizing a third party has the advantage of allowing them to verify the security of your code for compliance or customer requirements. It is advised that initial standards be created and followed because this typically doesn't happen until after the code has been created. The third party can then focus on adherence, verify it, and find other security flaws.

Phase 5: White/Gray/Black Box Testing

The planning of the security test includes the setup of the test environment. It helps with the need to schedule, monitor, and control operations related to setting up test environments when material procurement may have substantial lead times. The test team must plan and monitor environment setup tasks, install test environment hardware, software, and network resources, integrate and install test environment resources, obtain/improve test databases, and create environment setup scripts and test bed scripts. Regression tests and other tests are conducted, evaluation activities are carried out to prevent false positives and/or false negatives, security problems are documented via system problem reports, developer understanding of system and software problems is supported, and problems are tracked until they are resolved.

Phase 6: Determine Exploitability

Every flaw found during the testing stage of the SSDL should ideally be simple to resolve. The amount of work needed to fix a vulnerability might vary greatly depending on what caused it, whether it was a design flaw or an implementation issue. The risk posed by a vulnerability is significantly influenced by how easily it can be exploited. Using this data, you may decide how important it is to fix the vulnerability before moving on to other development needs like adding new features and taking care of security issues. Determining a vulnerability's exploitability involves weighing five factors:

- The access or positioning required by the attacker to attempt exploitation
- The level of access or privilege yielded by successful exploitation
- The time or work factor required to exploit the vulnerability
- The exploit's potential reliability
- The repeatability of exploit attempts

CHAPTER 3

State Of The Art

This chapter aims to analyze the current state of the art of security testing to identify the existence of different methodologies used by researchers. Another aspect is to compare the gap between researchers and companies to understand whether the figure of the "security tester" exists and how security aspects are managed.

3.1 Why Is Software Testing Different with respect to Security Testing?

Software failures typically occur spontaneously in the real world but software security aims to make software behave correctly in the face of malicious attacks. It should be no surprise that most literature on software testing only addresses what occurs when the software fails, regardless of design. Therefore, the existence of an Intelligent adversary determined to compromise the system distinguishes software safety from software security [16] Security is a risk management exercise; it is always based on the information and services being protected, the abilities and resources of the adversaries, and the costs of potential assurance remedies. The identification of potential security issues and their effects can be aided by risk

analysis, particularly at the design level. Software risks can then help direct software security testing after being identified and ranked. A flaw that an attacker can use against you is called a vulnerability. There are numerous types of vulnerabilities, and researchers in computer security have developed taxonomies of them. Bugs at the implementation level and design flaws are the two main types of vulnerabilities. Although bugs are typically easier to exploit, attackers generally don't care whether a vulnerability is caused by a flaw or a bug. The idea of which vulnerabilities actually matter is changing as attacks become more sophisticated. Timing attacks, such as the well-known race condition, were once thought to be exotic but are now quite common. In a similar vein, trampoline-based two-stage buffer overflow attacks were once the purview of software scientists but are now common in zero-day exploits. Although they are the most common and serious defect category, design-level vulnerabilities are also the most difficult to manage. Unfortunately, it takes a lot of knowledge to determine whether a program has design-level vulnerabilities, which makes finding such flaws not only challenging but also particularly challenging to automate. The handling of errors in object-oriented systems, object sharing and trust issues, unprotected data channels (internal and external), incorrect or absent access-control mechanisms, a lack of auditing/logging or incorrect logging, and timing and ordering errors are a few examples of design-level issues (especially in multi-threaded systems). Security risks are almost always caused by these flaws [16]. Numerous lists show the known security flaws and the most prevalent ones. It typically takes knowledge outside of "general" software testing techniques to fully understand and test for these issues. Indeed, just as testers should not be viewed as security experts by default, neither should developers be [17]. What qualifies someone as a "security expert" is a topic for the next section, but generally speaking, we can extend the adage that a good developer does not equate to someone who is knowledgeable about security [18] to also include the test developer. Though it is obvious that security must be taken into account at every stage of the software development life cycle [19].

3.2 Security Testing Knowledge Domain

What are the knowledge domains needed for security test engineers? Someone in this position needs to be an experienced programmer with test development experience, but the nature of the work also calls for knowledge of and access to additional skill sets in the fields of security and cryptography. Barnum et al. [20] divide such requirements into the following three major categories of expertise and knowledge:

- **Development**: a capable programmer with at least one high-level programming language.
- **Testing**: knowledge about how to formulate test case specification, test cases, general testing techniques, code review, manual inspection, etc.
- **Cryptography**: knowledge concerning security principles, e.g., types of algorithm, knowledge on the security properties of implemented solutions, etc.

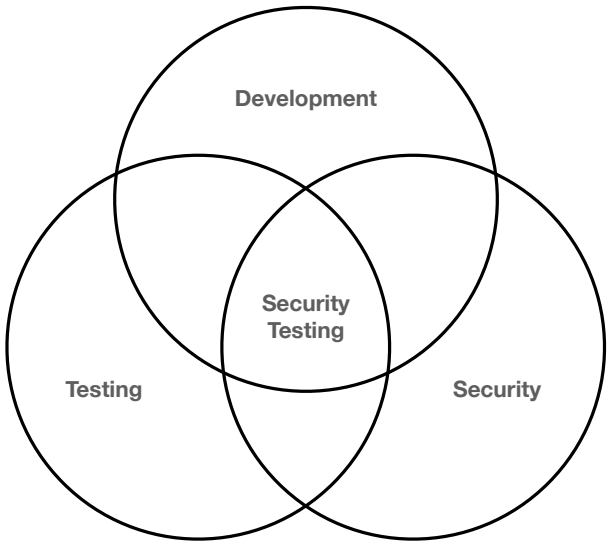

Figure 3.1: Areas of knowledge required for security testing

Figure 3.1 shows the intersection between the three categories in order to create a "Security Testing Engineer". Instead, Potter et al.[16] says that the question of "*Who*" should take part in security testing has two answers. Functional security testing can be carried out by conventional testing companies. One common functional testing

task is verifying that access control systems function as promised. However, risk-based security testing will be more challenging for conventional QA specialists. The issue is one of knowledge. First, because the designer must think like an attacker, security tests (especially those leading to full exploits) are challenging to create. Second, there is a difficulty with visibility because security tests do not frequently result in direct security attacks. A security test could produce an unexpected result that necessitates a more in-depth examination on the tester's part.

Kreeger [21] surveyed 300 undergraduate students in the United Kingdom, to understand whether there is a direct connection between the lack of knowledge in security testing and what is being taught at university. The results were not surprising because, in the first year of university, all students stipulated a programming course, and 80% of them mentioned the basis of software testing without security testing principles. During the second year, 60% of the students mentioned software testing activities, with particular attention to black-box and white-box testing. In the third year instead, 45% of the students attended an optional course in Security and Cryptography. So, the results showed that the group of students surveyed had a lower understanding of security-related tests. It can be useful whether universities start to contribute to the knowledge of security testing in the same manner as software testing.

3.3 Factors Affecting Software Security Testing

Nowadays, Software Testing practices use the *Unified Modeling languages* (UML) to move from the **Specification** (what task the application have to do) to the **Test Cases** (verify that specification works as expected). However, some bug types frequently escape testing [16]. Since many of these flaws do not involve traditional specification violations, the application may still function as intended while also carrying out some additional, unspecified tasks. Since testers design test cases to look for the presence of some correct behavior rather than the absence of additional behavior, bugs like these would inevitably escape the majority of automated testing. For this reason, Thompson in [22] highlights the so call "*side-effect behavior*". A typical functional test case would look like this: "*When input A is applied, check if result B is present*". But

what if, in addition to generating outcome B, the program also executes another operation (let's define say C)? Testers would probably notice C if it were something obvious, like an unexpected dialog window popping onscreen. However, if it were anything more covert, like creating a file or opening a network port, testers might not catch it, and it could happen frequently throughout testing undetected. For example, administrator users can run RDISK in Windows NT 4.0 which helps to create an Emergency Repair Disk for a machine. More in specific, when the users want to run the software, they create a backup of machine information, including Windows Registry. RDISK during its execution creates a temporary file and includes universal read permission—which means that every guest user logged, could read sensitive data about the system's configuration and could use this information to perform an attack.

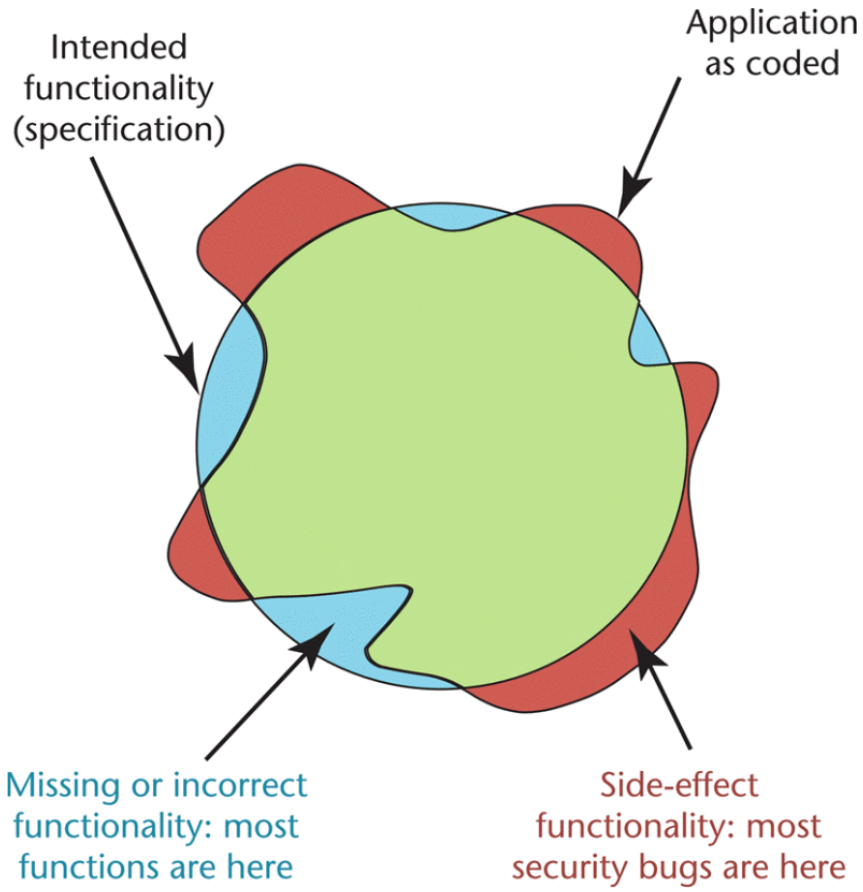

Figure 3.2: Thompson's side-effect of software vulnerabilities [22]

Thompson uses the amorphous shape in Figure 3.2 to show the side effect behavior of most software vulnerabilities. In particular, there are two relevant elements:

- **Circle**: that represents the application's intended functionality, usually defined by the specification.
- **Amorphous shape**: superimposed on the circle, represents the application's actual implemented functionality.

It is important to note that in a parallel world the two elements overlap if the specification matches the coded application perfectly, but in practice, this is difficult to achieve. As we can see, imperfect matches create two different types of spaces: the first one that does not exceed the specification (shown in blue in Figure 3.2) and the second one that goes beyond the intended functionality (shown in red in Figure 3.2) creating the side effect mentioned by Thompson. The mismatch can be reduced by taking security and especially security aspects into account early and in all phases of the software development life-cycle.

3.4 Development Frameworks for Security Testing

Starting from the description provided in Section 2.3.1, many companies have built their secure development frameworks based on the software life cycle, modeling it in a way that everyone can benefit of. The frameworks that stand out most in the literature are:

- **Microsoft Secure Software Development Life-cycle** (MS-SDL)
- **Open Software Assurance Maturity Model** (OpenSAMM) from OWASP

3.4.1 Microsoft Secure Software Development Life-cycle (MS-SDL)

Since 2004, Microsoft uses and promotes the Microsoft Security Development Life-cycle (Microsoft SDL)[23] as a software development method to save software maintenance costs and improve the software's dependability on security-related problems. To improve software security, Microsoft SSDL entails changing a software development organization's processes and including security-related measures. There are seven Development phase of MS-SDL activities [23], represented in Figure 3.3:

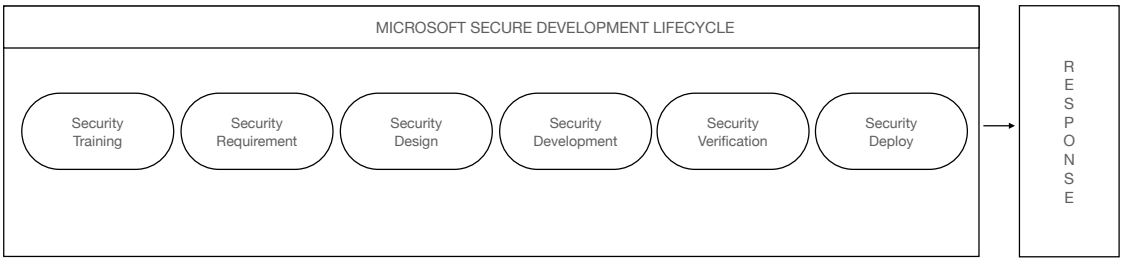

Figure 3.3: Microsoft Secure Development Life-cycle phases

- **Training**: Get informed about security basics and recent trends in security and privacy.
- **Requirements**: Analyzing security and privacy risk, defining quality gates.
- **Design**: Threat modeling , attack surface analysis.
- **Implementation**: Specifying tools, enforcing banned functions, static analysis.
- **Verification**: Dynamic/Fuzz testing , verifying threat models and attack surface.
- **Release**: Response plan, making final security review, releasing archive.
- **Response**: response execution.

Windows Vista was the first operating system to benefit from the MS-SDL, and Figure 3.4 indicates security improvements derived from the SDL when compared to Windows XP [24], showing a 45% decrease in vulnerabilities. Given the utility microsoft provides some guides on how to learn how to use its framework properly, called the "Developer starter kit".[1]

3.4.2 OWASP OpenSAMM

The Software Assurance Maturity Model (SAMM) [25] is an open framework for assisting organizations in developing and implementing a software security strategy that is adapted to the particular threats the organization faces. The tools offered by SAMM will help with:

- Evaluating an organization's existing software security practices

[1] `https://bit.ly/3Bs2U9z`

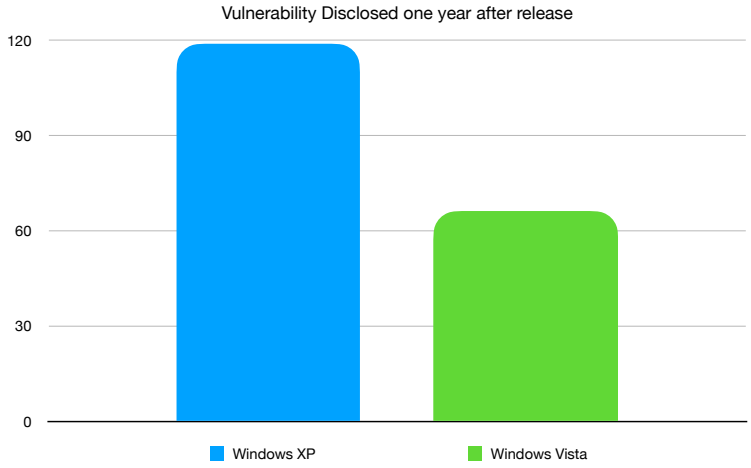

Figure 3.4: One Year Vulnerability Report

- Building a balanced software security assurance program in well-defined iterations
- Demonstrating concrete improvements to a security assurance program
- Defining and measuring security-related activities throughout an organization

In order to be used by small, medium, and large organizations using any type of development, SAMM was defined with flexibility. This model can also be utilized across the entire organization, for a specific line of business, or even for a single project. Beyond these characteristics, SAMM was created according to the following guidelines:

- An organization's behavior changes slowly over time - A successful software security program should be specified in small iterations that deliver tangible assurance gains while incrementally working toward long-term goals.
- There is no single recipe that works for all organizations - A software security framework must be flexible and allow organizations to tailor their choices based on their risk tolerance and how they build and use the software.
- Guidance related to security activities must be prescriptive - All the steps in building and assessing an assurance program should be simple, well-defined, and measurable. This model also provides road-map templates for common types of organizations.

The model's foundation is based on the core business functions of software development (see the second layer of Figure 3.5), with security practices linked to each. The three maturity levels defined for each of the twelve security practices serve as the model's foundation. These activities define a wide range of activities that an organization could undertake to reduce security risks and improve software assurance. As an open project, SAMM content must always be vendor-neutral and freely accessible to all.

OpenSAMM uses the fundamentals of the Capability Maturity Model, which is a framework for establishing the maturity level of processes within a company [26]. In this model, a value between '0' and '3' is assigned to the maturity level of security operations. '0' indicates that the operation is not used, while '1' indicates that there is no systematic approach but a basic level of security application at an organization. '2' indicates that the operation has reached a sufficient level of maturity within an organization. Finally, '3' indicates that the operation is perfectly applied in an organization. Project or time-based audits, according to this model, could improve an organization's security level [25].

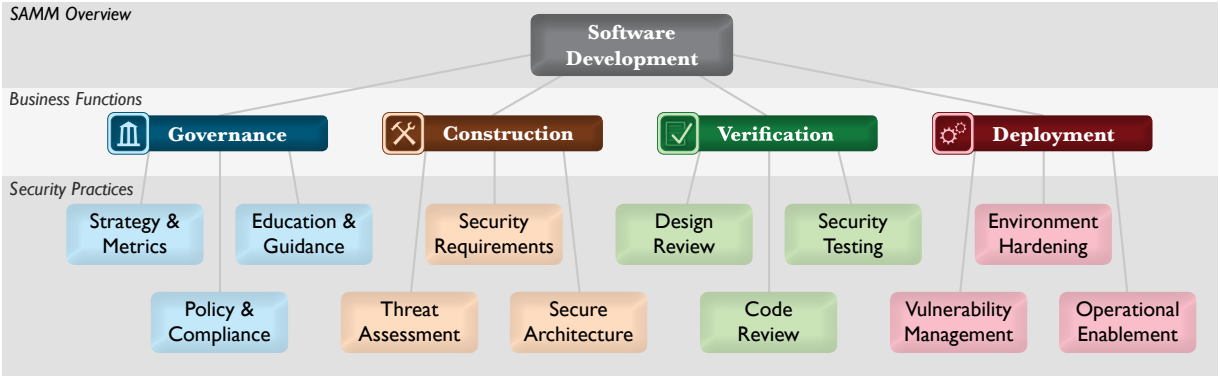

Figure 3.5: OpenSAMM Structure

3.4.3 Comparison Between MS-SDL and OpenSAMM

Kara in [27] compared the security properties of Microsoft SDL and OpenSAMM, according to the security properties of the models as given in Table 3.1. Both frameworks focus on secure software development.

To stay informed about security basics and recent trends in security and privacy,

Table 3.1: Security Property differences

Security Properties	MS-SDL	OpenSAMM
Security Training and Awareness	Yes	Yes
Physical and Logical Security	No	No
Secure configuration Management	No	No
Law, Policy and Procedure	**No**	**Yes**
Threat Modeling	Yes	Yes
Risk Analysis	Yes	Yes
Security Requirement Definition	Yes	Yes
Security Architecture	Yes	Yes
Secure Design	Yes	Yes
Source Code Analysis	Yes	Yes
Vulnerability Analysis	Yes	Yes
Security Verification	Yes	Yes
Vulnerability Management	Yes	Yes
Secure Development Technique and Application	Yes	Yes
Operational Environment Security	Yes	Yes
Secure Integration with Peripheral	Yes	Yes
Secure Delivery	**No**	**Yes**

all members of software development teams must receive appropriate training. In particular, person who creates software should receive regular security training. Microsoft SDL and OpenSAMM evaluate employee education and awareness. Physical and Logical Security as well as Secure Configuration Management are are not directly addressed on both the framework. An organization may be subject to a wide range of legal, policy, and compliance requirements. These requirements have an impact on the organization's software or hardware products, either directly or indirectly. Although Microsoft SDL does not directly address law, policy, and procedure compliance, OpenSAMM does. Threat modeling is used to identify significant security risks. This approach allows development teams to consider, document, and discuss the security implications of designs in a structured operational environment. Threat modeling allows security issues to be considered at the asset or application level. Threat modeling is a collaborative effort for developers and testers, and it is the primary security analysis task performed during the software design phase. During the development phase, OPENSAMM performed application-specific threat modeling. Moreover, threat modeling has been carried out since early phase of Microsoft's SDL. Risk analysis is the management phase in which possible risks to the software are identified. So, at this phase, a contingency and avoidance plan is drawn up for each risk. Both frameworks use these approaches. Security requirements, which provide techniques, methods, and standards for solving this task in the information system development cycle, are a critical part of the software development process for achieving secure software systems. The software development process should be repeatable and systematic. This approach ensures that the set of requirements obtained is complete, consistent, understandable, and analyzable by the various actors involved in the system's development. The designer of all frameworks considers a threat, assumption, and organizational security processes to be related and checked with IT and non-IT security requirements. Early in the software development life cycle, secure architecture and design must be considered. If this is not taken into account, security flaws at the design level may occur. Security architecture and design are required components of software engineering to ensure proper attention at all appropriate stages of the software development life cycle. Source code analysis is a technique that is used for multiple purposes it originated as a technique —both

static and dynamic to identify possible bugs within the code. Later it is also applied to security contexts, to look for the presence of known vulnerabilities, using special tools, where the results are checked by experts to eliminate possible false positives. Steps ranging from vulnerability analysis to secure integration are steps derived from the use of source code analysis. The last phase, secure delivery deals with the delivery of the final product directly to the user. Delivery requirements must be defined to provide assurance during distribution of the product to the user. Only Microsoft SDL deal with secure delivery.

3.5 Security Testing Methodologies in Practices as Research Topic

From what is understood by analyzing the literature, especially from the study of Bruce et al. [22] , it is unclear who performs security testing and whether there is such a figure as a tester or software developer. More specifically, it is unclear what security practices should be considered in real business contexts, for instance:

- Are we sure that all IT companies use frameworks based on a secure development life-cycle?
- How are security testing procedures organized during software development?
- How much does the size of the company affect this choice?
- Which of the many frameworks do they decide to follow?
- Is the entire framework process followed entirely or are only certain steps taken?
- Would it be useful to start with one of these frameworks and expand it according to business needs and eventually develop its own security standard?
- How and by whom is the entire process managed?

There are still many unanswered questions in the literature, some of which could be answered through this study. So, it would be useful to create an empirical study

aimed at understanding how different business frameworks deal with security issues. The study can then be enriched by subsequent focus groups or individual interviews to gain further details.

CHAPTER 4

Design

This chapter illustrates the research questions and the research strategies used to achieve the main objective.

4.1 Research Questions

4.1.1 Main RQ - Business Perception on Security Testing

Resuming the discussion concluded in the last section of the previous chapter on state-of-the-art reflections, we discuss possible research insights in this field. The entire empirical process is aimed at answering the following Research Question (RQ):

Main RQ

How is the concept of security testing being used in business contexts?

This macro question aims to provide the research community with an overview of current security practices adopted by practitioners. The macro question is divided into different research sub-questions that guide the entire empirical process.

4.1.2 RQ_1 - Security Tester Figure

RQ_1

Who might perform security testing in business company?

From the analysis of the state of the art, we observed that the concept of security testing can be implemented in different ways and contexts. However, most researchers agree with its definition, i.e., "***Security Testing can be defined as the set of activities aimed at predicting and finding potential security problems in a software system, with the final goal of increasing the system's robustness.***". Regarding the execution of security practices, there are different opinions on the skills a security tester should have: some researchers believe that a software security officer should have a background of specific security knowledge—e.g., encryption algorithm, types of algorithms, an evaluation of the security properties of the implemented solutions, etc. — while others argue that a security tester should have basic knowledge in software development (knowledge of at least one high-level programming language), basic knowledge of software testing, and basic knowledge of cryptography. In this work's context, we tried to validate the second line of thought, consequentially adapting our research methods to identify which professional figures perform security testing in practice.

4.1.3 RQ_2 - Who Is Responsible for the Application of Security Frameworks?

RQ_2

Are we sure that all IT companies use framework based on secure development life-cycle?

One of the right ways to develop security testing is to rely not only on the software life cycle but also on the secure software life cycle. As seen in the previous chapter, numerous frameworks have been introduced over the years that can guide many security testing activities [28, 25]. Nevertheless, these frameworks are relatively new, and—for that reason—their use in the professional world has not yet been explored. Our second research question aims to shed light on such limitations.

4.1.4 RQ_3 - The organization of security testing in the company

RQ_3

How are security testing procedures organized during software development?

To answer this question, an individual interview could be conducted with one participant from each company. It could be useful to highlight what techniques are used in the absence of a security framework, or highlight the processes if they are used. This would allow all aspects of security testing to be explored in depth and used as a model for comparison among the companies themselves.

4.1.5 RQ_4 - Teamwork

RQ_4

Who is the Security Testing process coordinated by? How often is security testing conducted in groups?

To conclude the analysis on security testing, we wanted to know whether security activities can be carried out in groups of people—in a "pair-programming" fashion—or whether a solitary activity can achieve better results. In addition, it would be useful to understand whether there is a figure comparable to the project manager, who coordinates and manages the entire security testing procedures, interfacing with project managers or top managers.

4.2 Research Methods

In order to obtain useful information to achieve the research objectives described above, it was decided to design the data collection steps in the form of a survey to be presented to experts in the field. In addition, an individual interview was designed to obtain useful details to answer the research questions.

4.2.1 Overview of the Research Method

Conducting a qualitative study to investigate security testing activities used in different companies can have a triple advantage. First, through questioning domain experts, it will be possible to extract valuable pieces of information also for practitioners and not only for researchers. Second, Such an approach could be the starting point for future and more specific work—more related to the specific strategies and training approach in the security testing context. Last, the extracted knowledge could be inspiring and valuable also for small teams and/or companies with insufficient "budgets" to conduct in-depth analyses. Our methodology is based on 4 steps: (i) collecting data; (ii) filtering data according to each participant's knowledge; (iii) having a semi-structured interview with one of the companies; (iv) gathering information and answering research questions

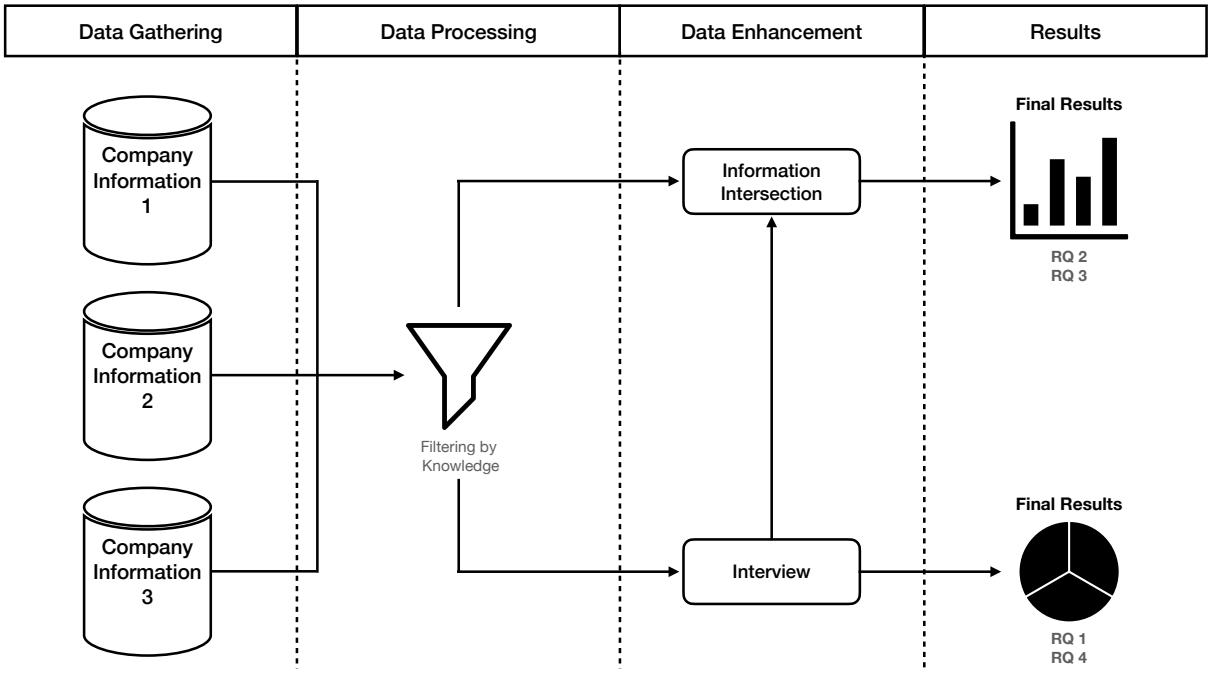

Figure 4.1: Research Process Overview

4.2.2 Survey Structure

From an initial review of the literature, security testing appears to be a technique that few employ following "standards." For this reason, it was decided to create a small survey that could encapsulate all the main and necessary information, making

it effective enough to meet most of the research objectives. The design takes the structural directions provided by Andrews et al. [29], which are:

- The formulation of multiple-choice or scaled (numerical or qualitative) answers, to give a more analytical character to the data that can be extracted from the questionnaire responses;
- The use of clear, unambiguous and coincidental vocabulary in order to eliminate ambiguity about the meaning of the question;
- Specifying what are the objectives of the Survey are and making it clear from the outset that the information taken will not be used for any other purpose;
- Gathering information about the survey participants, which is useful in strategically categorizing the available data;
- Ensure respect for privacy by specifying that the data will be treated in anonymous form from the point of view of analyzing and publishing the results, without making any reference to who provided the responses;

Based on the reported principles, the research survey was formalized into 5 sections, including 3 main sections for the study. In order to ensure consistency of content, there are three discriminating questions (based on the literature), which have been used to verify that the respondent has the necessary skills to complete the questionnaire accurately. To carry out the Survey, we chose to use the *Google Form* platform which allows to (1) formulating various types of questions according to different inquiry needs and (2) divide the questions into different sub-sections that are consistent with what is designed.

Given the short composition of the survey, it was decided to adopt the division into sub-sections, where:(1) The first section shows a brief description of the research and provides definitions on security testing; (2) the second section composes the discriminatory questions; (3) the third section is concerned with the use of security testing in real business contexts; the last two sections deal with business and personal information. The duration of the questionnaire was set at around 10 minutes, thus

avoiding drops in attention. The participants recruited for the questionnaire were employees of 3 different companies, having different business models and customers. It is important to note the questionnaire was duplicated for each participating company. This also makes it easier to validate the responses.

One of the biggest risks is creating a lengthy survey, as this can easily lead to a lapse in attention, making the responses collected invalid. One possible plan to avoid this is to conduct a pilot test. The test was conducted with master's students from the University of Salerno and with young developers from different companies. Both were attending a degree program at the University of Salerno. The participants were asked to fill in a copy of the electronic survey to estimate the running time and to identify the presence of unclear questions or definitions. Participate requirements for the pilot test are:

- Know at least one high-level programming language.
- Have participated in a course or work in the security field.
- General knowledge of software engineering

All answers from the pilot test are not considered in the evaluation of the results. The results of the pilot test were satisfactory: the 5 participants—including one with dyslexia—completed the survey within the allotted time. The participant with dyslexia completed the survey, taking longer but staying within the allotted limits. Participant feedback was consistently positive, with the exception of some questions that were later reworded.

Figure 4.2 summarizes the general structure of the survey. In the following, each survey part has been deepened in detail.

Survey Introduction

Before beginning with the basic parts of the survey, it is useful to describe the first section devoted to describing the research problem, providing the purposes of the study itself and some non-formal definitions of security testing. Information on the duration of the survey and the handling of sensitive data is also provided.

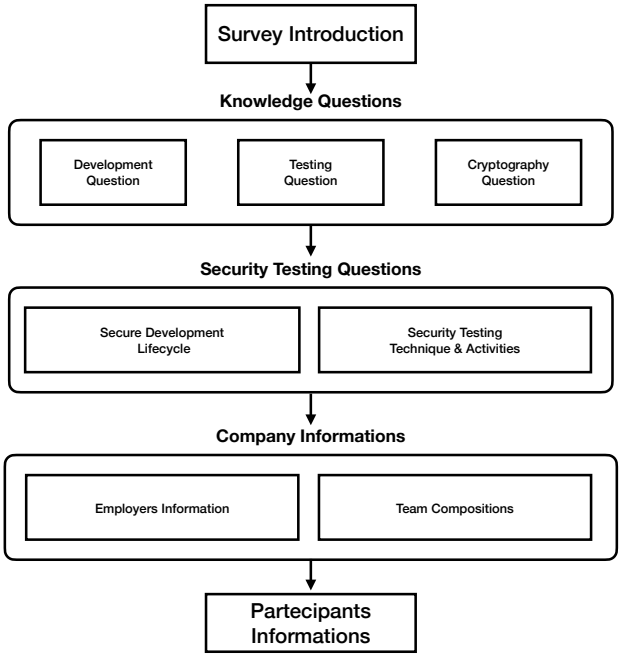

Figure 4.2: Survey Structure

Knowledge Questions

The second section is one of the most important as it aims to quantify participants' knowledge. Users were asked three questions, each with different targets.

- Knowledge about Software Development.
- Knowledge about Software Testing.
- Knowledge about Cryptography.

Table 4.1 resume the entire section with all the characteristic. The first question shows an image with a few lines of Java code reported in Listing 4.1, which represents a simple recursion on tree nodes. In this context, the programming language does not play any discriminating role; in fact, all companies recruited for the research study base their foundations on high-level languages and object-oriented languages.

Listing 4.1: Development Question

```
1 public void func(Tree root) {
2     System.out.println(root.data());
3     func(root.left());
4     func(root.right());
5 }
```

The second question is about software testing, specifically, Figure 4.3 shows a code snippet and its representation using the Control Flow Graph (CFG). With this question, we want to find out whether the participants are able to read the code and find the infeasible path by the use of a Control Flow Graph. This would provide evidence that the participant is capable of deriving test cases from code.

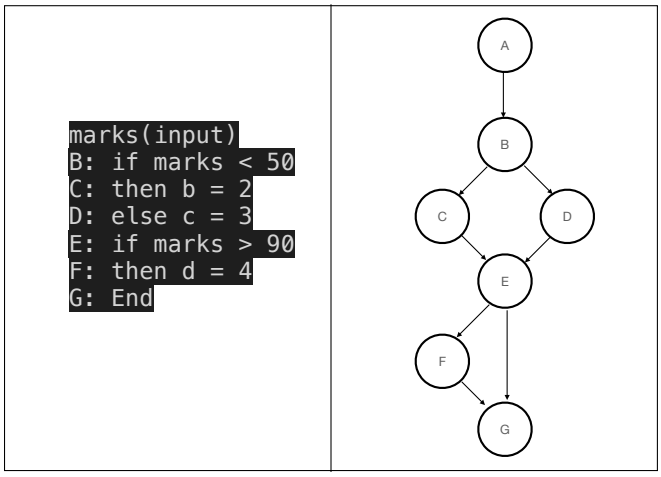

Figure 4.3: Code Snippet and Control Flow Graph

In the end, Figure 4.4 shows the cryptography question. We want to understand whether the user has at least basic knowledge of one of the most widely used cryptography methods. The Hash function can be used in many contexts; one of its uses involves saving passwords to a database.

As mentioned earlier, a participant must answer correctly at least two of the three questions submitted to be accepted, thus creating a scoring ranking as follows:

- **Partecipant accepted**: all answer are correct (3 points).
- **Partecipant partially accepted**: two of three correct answer (2 points).
- **Partecipant discarded**: one or zero correct answer (1 point / 0 point).

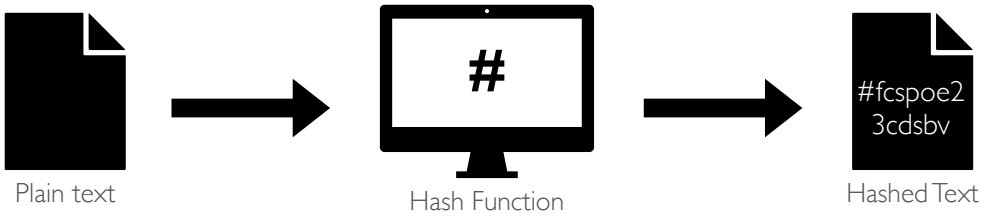

Figure 4.4: Hash Function

Table 4.1: Questions characteristics

Question	Type	Mandatory	Correct Answer
What does this piece of code do?	Multiple choice	Yes	Preorder traversal
What goes wrong with code?	Multiple choice	Yes	1 infeasible path
The Hash function is...	Multiple choice	Yes	One way

Security Testing Questions: How to approach to Security Testing?

This section contains all the main questions for investigating security testing in companies. Ad-hoc questions have been built to clarify what organizational procedures and methodologies are used to develop secure software. Questions such as:

- **Subgroups**: As a first question, it has been 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 developers, testers, security specialists, UX/UI specialists, etc. 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. A division into work groups is useful to have people with different backgrounds and skills working together to achieve business goals. Age, nationality, ethnicity, religious background, personal history, professional background, skills, gender, or political preferences may differ. These distinctions can help companies discover creative solutions and innovative methods to improve operations and achieve operational goals.

- **Security Aspects**: participants are asked whether the entire company used to manage security aspects.

- **Secure Development Life Cycle and Framework**: This question is crucial. Participants are asked if they are aware of secure development life-cycle, but especially if they apply frameworks such as the one created by Microsoft (MS-SDL) or any other available framework. All affirmative answers will have a huge impact on the research, as it will be possible to organize focus group sessions, or individual interviews aimed at clarifying the entire development process.

- **Security Management in the Software Life Cycle**: 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. Also, it may be useful to understand whether the frameworks specified earlier, are full executed or partially.

- **Testing Methodologies**: By analyzing the responses, it could assume the possible techniques applied. In fact, if a participant states that requirements analysis plays a key role in security, then it is likely that “Architecture security review" techniques are used. This can be learned from focus groups or single interviews to mitigate what methodologies are covered in the literature and whether they are actually used or whether some are preferred to be used over others and why.

- **Types of Applications Developed**: Useful to understand whether frameworks or individual techniques applied to the life cycle, are changed as a result of the type of application to be developed.

- **Security Testing in Groups**: The level of utilization of testing activities in groups of people.

This section of the Survey touches on a whole range of aspects that deserve to be investigated possibly with other focused studies, however, certainly investigating

these concepts, is certainly a relevant starting point for security testing. Table 4.2 report a summarization of the questions proposed.

Company & Participant Information

The last two parts of the questionnaire refer to the collection of company characteristics and the collection of participant characteristics. The company characteristics are useful to compare and get a complete picture of the different companies that participated in the survey. The participant details are useful for obtaining 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 results of the survey with them if they are interested. Table 4.3 shows a summary of all questions for collecting company information and Table 4.4 shows a summary participant information.

4.3 Survey Recruitment and Dissemination

The study aims to obtain information from professionals working in the field of enterprise software development. The professionals we are interested in are:

- Developer
- Software Tester
- Security specialist
- Senior Manager (Chief Executive Officer, Chief Technology Officer, Project Manager etc)

4.3.1 Participants Recruitment

To recruit participants, we planned to contact several companies in the IT sector with different development and size goals. The companies included in the recruitment plan are:

- Zucchetti S.p.A.[1]

[1] https://www.zucchetti.it/website/cms/home.html

Table 4.2: Questions Report

Question	Type	Mandatory
In the context of your company (or team), is it possible to identify subgroups of people who deals with specific activities? Or you don't need any subgroup separation?	Multiple choice	Yes
Are security aspects managed in the context of your company?	Multiple choice	Yes
Have you ever used a Microsoft Secure Development Life-cycle or any other security development lifecycle?	Multiple choice	Yes
In what software development phase are you usually apply software security methodology?	Checkbox	Yes
Based on Security Testing for each software development life-cycle (SDL), which of these techniques do you use?	Multiple Choice Grid	Yes
Do these types of techniques change according to the application type to be developed?	Multiple Choice	Yes
In your experience, how often are security testing activities carried out in groups?	Linear Scale	Yes

Table 4.3: Company Information

Question	Type	Mandatory
Do you work for a multinational company?	Multiple choice	Yes
How many employers are there in the company?	Multiple choice	Yes
How many members are there in your team?	Multiple choice	Yes

Table 4.4: Participant Information

Question	Type	Mandatory
How old are you?	Free Text	Yes
What is your gender?	Free Text	Yes
Nationality	Free Text	Yes
What is the highest degree or level of school you have completed?	Free Text	Yes
What role do you play within company?	Multiple Choice	Yes
How many years of experience do you have in that role?	Multiple Choice	Yes
How many years of experience do you have as Security Expert?	Multiple Choice	Yes
Would you like to learn about the result of this study or to be contacted to participate in a follow-up interview/focus group on this topic? If yes, please write down your email.	Free Text	No

- SAP S.p.A.[2]
- IT.Svil S.r.l.[3]

This research study involves experienced industry professionals who are busy with their work, so it was not possible to set a minimum number of responses that should be collected. This condition was considered an important problem, which will be explained and resolved by using the next section research technique. In addition to the companies, the survey was distributed on various social media such as Twitter and Reddit with less satisfactory results. The survey was released on different dates: IT.Svil starting July 7, SAP a starting July 13, 2022, and Zucchetti starting July 27. However, the timeline for completing the survey was extensive, the survey was available until September 1, 2022. On the same day, the Google form was disabled via a corresponding function.

4.3.2 Single Interview

As we approached the closing date of the survey, we reached a reasonable amount of responses (26): while the number might look low, it is worth remarking that our study targets security experts - involving them is challenging and, therefore, relying on the opinions of 26 recognized experts is already a valuable result. Nonetheless, we did not limit ourselves to the survey: in an effort of providing more insights into the methods applied to perform security testing in practice, we conducted additional semi-structured interviews with some of the survey participants. We decided to recruit the *Chief Technology Officer* (CTO) of Zucchetti S.p.A. as an expert in many areas, from software engineering to secure software development. More specifically, single interview research is defined as a single person carefully selected who contribute to open discussions for research [30]. In the context of our study, a single interview enables a joint discussion on current security testing practices based on software development life-cycle. During the meeting, the first author of the thesis 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

[2] `https://www.sap.com/italy/index.html`
[3] `https://www.itsvil.it/`

Table 4.5: Single Interview Questions

Question	Type	Mandatory
What types of "Secure Development Lifecycle" are you familiar with? If more than one is shown, then which one is used most?	Live Talk	Yes
If multiple approaches are used, how are multiple frameworks merged? Could the operations be redundant?	Live Talk	Yes
If they use one, based on what characteristic did you choose one over another?	Live Talk	Yes
How is the entire security testing process organized? For example, Zucchetti is in charge of the development of particular software. How does the security process begin? and who is in charge of carrying it out? Who is in charge of verifying that the entire process is carried out correctly?	Live Talk	Yes
Based on Security Testing for each software development life-cycle (SDL), which of these techniques do you use?	Live Talk	Yes
In your projects, do you prefer to carry out Security Testing activities in groups or individuals? Why? Have you noticed differences between the two approaches?	Live Talk	Yes
Reviewing the results of your questionnaire, I could see that the requirements acquisition phase does not undergo any security process, while the above frameworks recommend it. Why is this choice and what does it depend on?	Live Talk	Yes
Reviewing the results of your questionnaire, I could see that the maintenance phase does not undergo any security iteration, while the above-mentioned framework recommends it. Why is this choice and what does it depend on?	Live Talk	Yes
In the other phases of the life cycle (Software Design, Software Development, and Software Verification), what techniques do you use to fulfill security testing? Moreover, do these techniques vary depending on the type of application being developed? If not, why?	Live Talk	Yes
Which phase of the security life cycle or which methodology associated with it requires the most effort?	Live Talk	Yes
Further Comments	Live Talk	Yes

several existing frameworks in the literature. Two other components participated in the interview with the goal of transcribing as much information as possible. At the end of the meeting—with permission to record the interview using Microsoft Teams function—the author combined the information with the manual transcription of the entire interview. Therefore, additional questions were created and are summarized in the Table 4.5.

CHAPTER 5

Analysis of the Results

This chapter describes the results obtained from business interviews, analyzing how they use Security Testing techniques and answering research questions.

5.1 Data Cleaning

As mentioned in the previous chapter, a total of 26 survey responses were received: 20 responses from I.T.Svil, 5 responses from SAP, and 1 response from Zucchetti. The total number of participants accepted for each company is shown below:

- **I.T.Svil**: 13 participants, of which 3 achieved the highest score and 10 achieved a medium score.
- **SAP**: 5 participants, of which 2 with the highest score and 3 with medium scores.
- **Zucchetti**: 1 participant achieved the highest score.

In addition, nine of the 19 participants left their email contact for possible interviews or focus groups, giving them the opportunity to add important contributions that were not mentioned in the survey.

Before we started the data analysis, it was necessary to evaluate the consistency and integrity of the data. The techniques used are described in 4.2.2.

5.2 Data Analysis

After discarding inconsistent or unacceptable data, the data sample turns out to be ready for analysis. The analysis phase made use of statistical aggregation tools and the formulation of descriptive graphs. Among various alternatives, the author chose Python to model the information.

5.2.1 I.T.Svil S.r.l

The first data to be analyzed are those of I.T.Svil, an established computer engineering company based in Salerno and Bologna. The company constantly works on the organizational and administrative processes of healthcare companies, both public and private. In this case, all the participants came from Italy and came from Salerno.

Personal Information

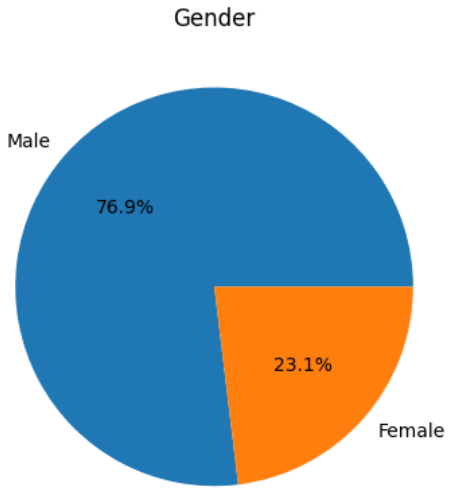

Figure 5.1: Gender Percentage I.T.Svil

From the Figure 5.1, identification of the gender of the participants is possible. The presence of this question was widely circulated in the design phase of the questionnaire, as it is information that should be considered highly discriminatory if not asked in the right way. For these reasons, the question allowed participants to respond via Free Text in which they were free to respond with "*I prefer not to answer*" or other options. The majority of participants, about 79% identified themselves as male, and the remainder identified themselves as female. No participants refrained from declaring their gender.

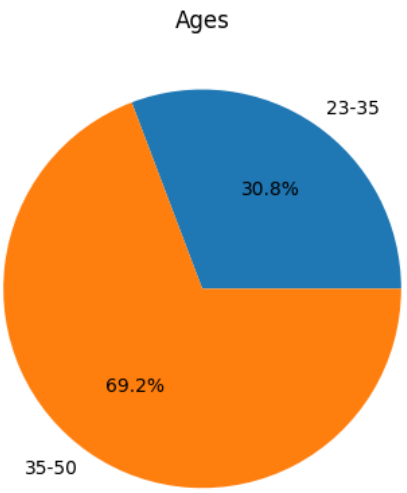

Figure 5.2: Ages Percentage I.T.Svil

Consider Figure 5.2 instead, it is possible to see that the participants range in age from 23 to 50. Specifically, 31% of the participants say they are between 25-35 years old, and 69% say they are between 35 and 50 years old. Next, we will have the opportunity to compare age with the type of role taken within the company.

Another information is the educational qualifications of the participants. From the beginning of the Survey design phase, educational qualification is considered an important but non-discriminatory factor to achieve good reliability of the information. Some participants did not have a bachelor's degree but instead had a high school diploma, but they demonstrated—responding correctly to the questions in the first section of the Survey— that they were knowledgeable about concepts of a complex

nature. Figure 5.3 shows that 70% of the participants had a bachelor's degree, and 15% in addition to a bachelor's degree had a master's degree. 7% have never earned a bachelor's degree.

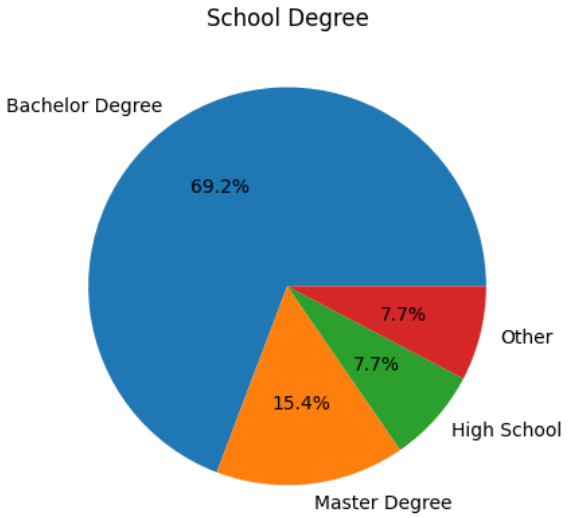

Figure 5.3: Education Percentage I.T.Svil

The professional positions of the participants are presented. Of course, the positions visible in Figure 5.4 are only a subset of the actual positions in business reality. However, the results show a certain homogeneity. Next, the professional positions of the participants are depicted. Admittedly, the positions visible are only a subset of the actual positions in business reality. However, the results show a kind of homogeneity. Most of the participants hold the ground of Software testers and Software Developers. Some participants hold —in addition to Software Developer or Software Tester—other positions, such as Project Manager, UX/UI Specialist, and Analyst.

Company Information

Analyzing Figure 5.5 we can see that 70 % of the participants say they work for a multinational company and the remaining 30% say the opposite. At first, the analysis of this data might confuse, considering that all participants work for the same company but have different conceptions. What we can imagine instead is the participants are located in other small companies that refer to the main holding company, namely

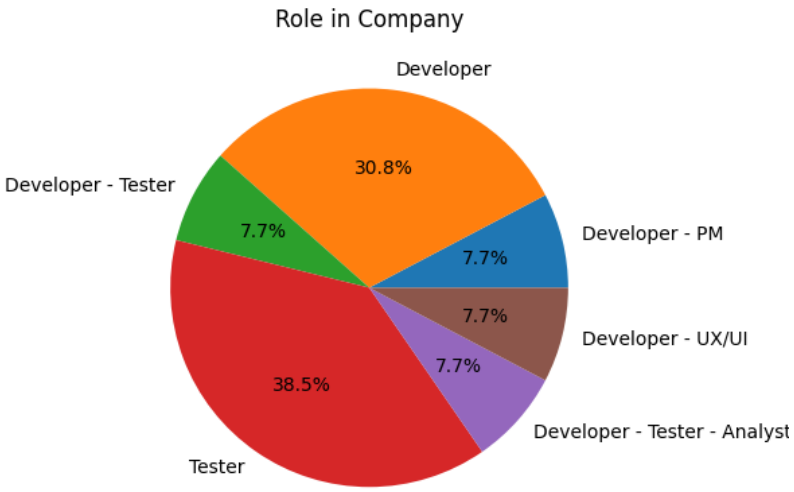

Figure 5.4: Company Role Percentage I.T.Svil

I.T.Svil. What they all have in common, however, relates to the number of employees in the company. Here we can refer to the main holding company, which together with the other dislocated companies reaches a large number of people. Figure 5.7 shows the number of employees.

Last but not least, it concerns the number of people who make up a team. Looking at Figure 5.7, participants say that teams also take on different sizes in relation to their tasks. About 46 percent report that the team they are part of usually consists of 6 to 10 people. The other 46% report that their conformation usually keeps to small team sizes of 1 to 5 people. Only 7% report working in larger settings with teams of 11 to 15 people.

5.2.2 SAP S.p.A.

In the following, we start analyzing the data of the company SAP. It is mainly engaged in the development of ERP management software for small, medium and multinational companies. The company currently has offices throughout Europe, but its headquarters are located in Walldorf, Germany.

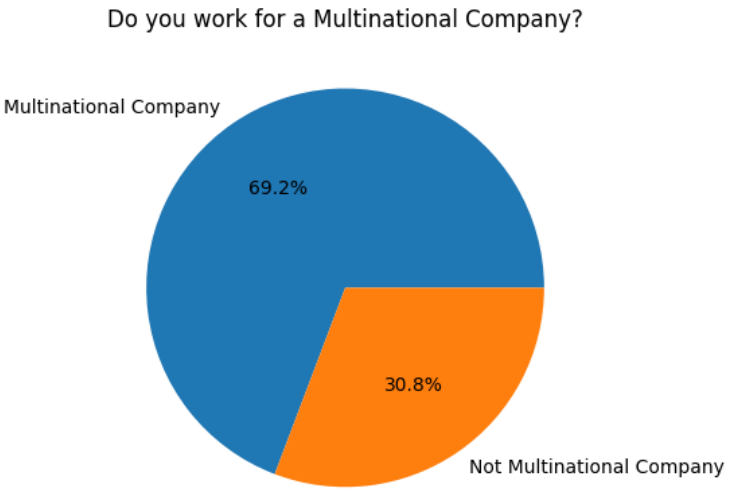

Figure 5.5: Multinational Percentage I.T.Svil

Personal Information

As mentioned in the previous paragraph, SAP did not receive a sufficient number of survey responses. However, since SAP is an information security company, we can get good input, including through interviews or focus groups. The same process used for I.T.Svil is used for SAP. We begin with participant information by starting with gender. Figure **??** shows that 100% of participants self-identify as male.

Because SAP has been a security company for many years, we can imagine that the most crucial roles are assigned to people of color who have gained some experience in the field over time. In fact, looking at the age representation of participants in SAP, we can see that most (except one) are in the older age range of 45-50 years old. Figure 5.9 shows that. Instead in Figure 5.9

By the same reasoning as before, it is easy to see that the course of study is also different. You tend to have people who, in addition to having an excellent background, have done and completed a course of study in computer science. Figure 5.10 shows that both 40% of the participants have been hired and have a three-year course of study. The other 40% have been hired and have a master's degree course of study. Finally, only 20% of the participants have obtained a P.h.D.

In this case, we have the prime example of how a safety company is truly focused

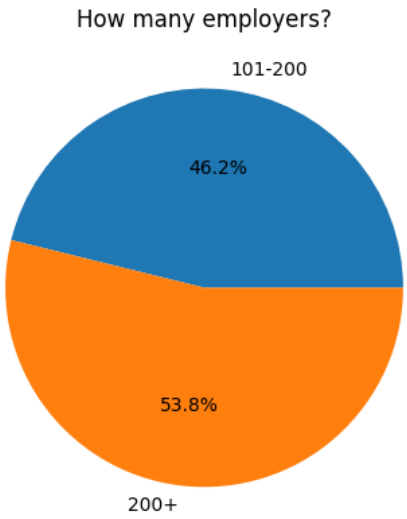

Figure 5.6: Employer in Company I.T.Svil

on safety: in the example of the company present, it was reiterated by the participants that safety practices are implemented them but without using the safe life cycle. Now we end up with a company that not only develops "classic Software," but also develops and controls security practices. Figure **??** shows that most of the participants —in addition to being good developers—attend to security practices.

Company Information

All of the participants, although they belong to geographically dispersed locations from each other say that SAP is an established multinational company and that the number of employees themselves proves this. We can precisely confirm this by looking at Figure 5.12.

5.2.3 Zucchetti S.p.A

Last but not least, let us analyze the participant data of Zucchetti S.p.A., that is one of the first software house founded in Italy, with branches all over Italy, Europe and the world. The areas in which Zucchetti works are mainly three: software for accountants, software for payroll and Human Resource (HR), and ERP management software.

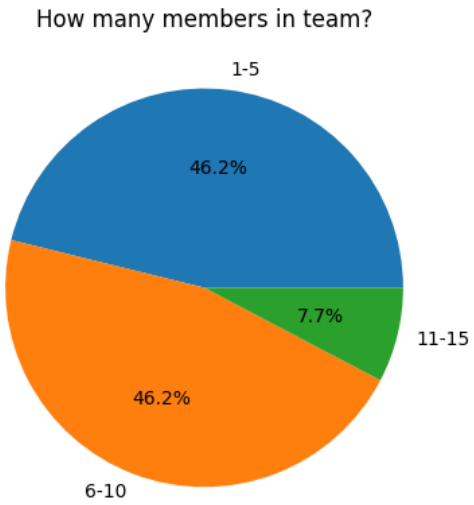

Figure 5.7: Number of Members within I.T.Svil Team

Personal Information

The only participant—and perhaps one of the most representative—Dr. Gregorio Piccoli, CTO of Zucchetti. Some of his responsibilities include:

- Monitoring new technologies and evaluating their potential as applied to products or services.
- The supervision of research projects to ensure that they bring added value to the company (especially from a security perspective).
- The technical evaluation of the potential of possible collaborations, acquisitions, or mergers of companies.

Company Information

From the information provided in the questionnaire, it appears that Zucchetti has a large number of employees. With a turnover of more than 675 million euros in 2018 and more than 350.000 customers, Zucchetti is the leading Italian software company. The group employs more than 5.500 people, with 1.200 of whom work exclusively in research and development.In 2021, the number of employees will increase to 8,000 and annual sales to more than 1 billion euros.

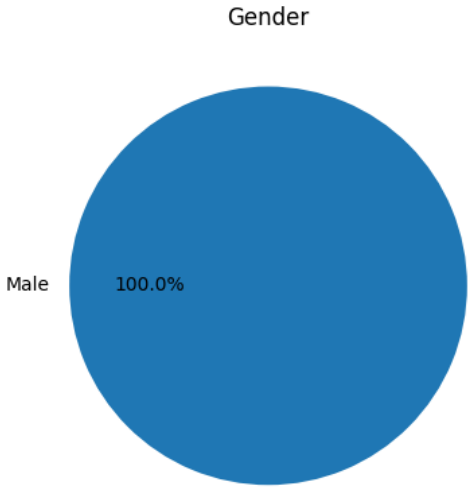

Figure 5.8: Gender Percentage SAP S.p.A

5.2.4 Security Testing Data Analysis

I.T.Svil S.r.l

I.T.Svil participants say they get the security aspects, however, they do not use a framework based on the security life-cycle. Initial analysis indicates that security is implemented and managed, but not according to specific protocols. After initial analysis, I.T.Svil participants indicate that the phase in which they do most security testing is design, development, and Verification as shown in Figure 5.13. What is changing, however, are the techniques used: Participants indicate that the techniques suggested in the survey for the design and verification phase are not the ones they actually use. For the development phase, on the other hand, they indicate that they use **Static source code analysis and manual code review**, i.e., analyzing the source code of the application to find vulnerabilities without actually running the application. In addition, most participants indicate that they do not know if the techniques change depending on the type of application being developed and do not allow the employees to test safety aspects in a group of people.

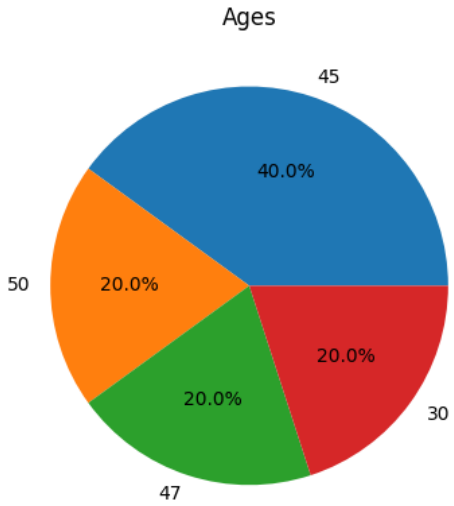

Figure 5.9: Ages Percentage SAP S.p.A

SAP S.p.A

SAP Participants, on the other hand, say that security aspects are managed using frameworks based on the security life cycle. Figure 5.14 also shows that all participants agree on the use of these activities across the software life-cycle, giving more importance to the phases: Design, Development, Verification, and Deployment. Some—also due to the different types of tasks within the company—affirm the importance of using preventive techniques directly in the requirements phase. Downstream organizations also affirm which security testing methods are used in each phase of the software life cycle. In the planning and design phase these are:

- **Architecture Security Review**, a manual review of the product architecture to ensure that it fulfills the necessary security requirements.
- **Threat Modeling**, a structured manual analysis of an application-specific business case or usage scenario.

For the Development phase:

- **Static Source Code Analysis and Manual Code Review**, the analysis of the application source code for finding vulnerabilities without actually executing the application.

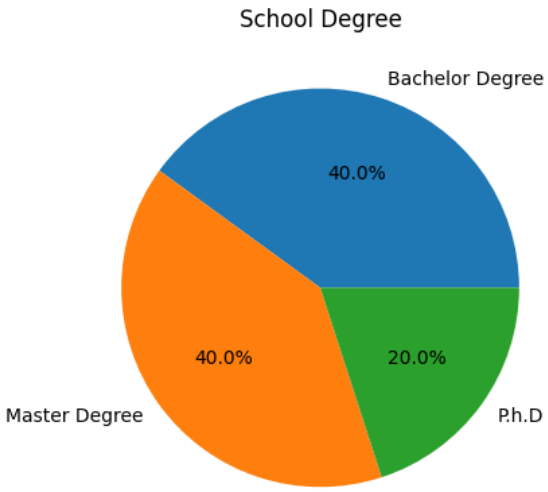

Figure 5.10: Degree Percentage SAP S.p.A

- **Static Binary Code Analysis and Manual Binary Review**, the analysis of the compiled application (binary) for finding vulnerabilities without actually executing the application.

For the Verification phase:

- **Automated Vulnerability Scanner** that tests an application for the use of system components or configurations known to be insecure.

One of the most important aspects is that security testing techniques vary as the nature of the application being developed changes. This leads to the assertion that although one must have knowledge in different contexts such as development, testing, and security, one must have a security-oriented background in order to decide what types of techniques to use for the type of application being developed.

Zucchetti S.p.A

Zucchetti also claims that security aspects are managed and that they are supported by Secure Development Life-cycle Frameworks. Figure **??** shows that the company uses most security testing techniques. The interview conducted with Dr.

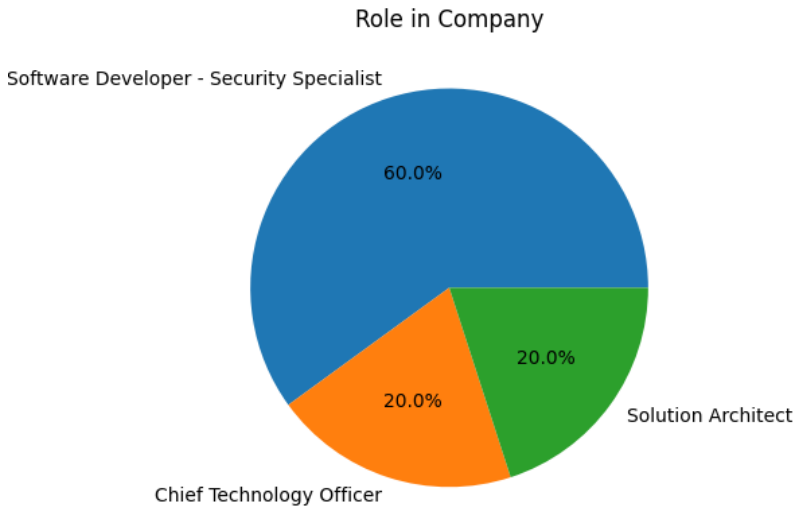

Figure 5.11: Role Percentage SAP S.p.A

Gregorio Piccoli, CTO of Zucchetti, shows that among the many frameworks available in the literature, the one used is not the one from Microsoft, but the one from OWASP. The main reasons for this are the degree of applicability through the Capability Maturity Model, which allows them to "train" new startups or newly acquired companies. However, he notes, "*Using the OWASP framework only makes sense if it is coupled with secure development by developers. Otherwise, we risk having the same limitations as software testing, i.e., that it's only used to prove the existence of problems, not their absence.*" Combining security testing with frameworks and secure development by developers brings the need to use DevOps and DevSecOps. "*At zucchetti, we prefer to build architectures that are tested as secure in and of themselves to give developers guidelines for software development without denying them the modulation they prefer.*" In addition, Dr. Piccoli says that in addition to working with OWASP and OWASP Italy, the company has formed its own Ethical Hacker Group to ensure dual prevention of software. Regarding the composition of the team, he says, "*Each development team has a security manager who is responsible for contacting the hacking team to communicate the start of security testing and to explain to the programmers the nature of the problems encountered.*"

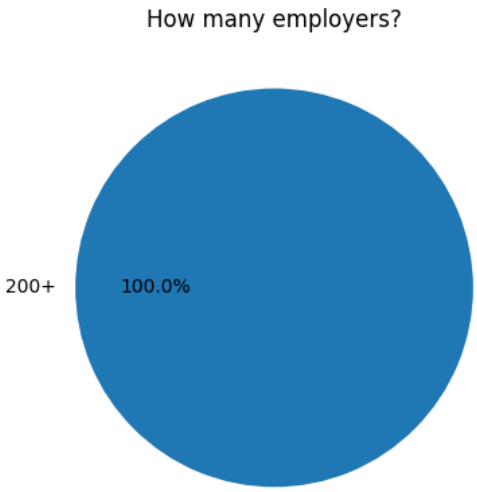

Figure 5.12: Gender Percentage SAP S.p.A

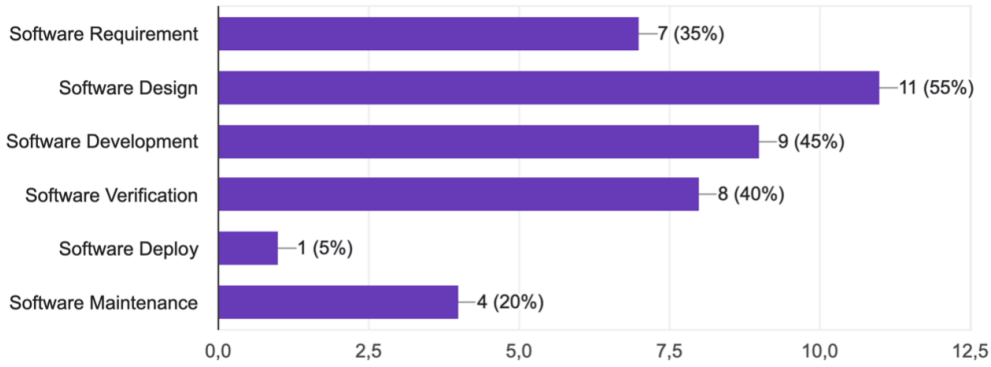

Figure 5.13: Security Testing Activities in Software Development Life-cycle for I.T.Svil

5.2.5 Research Questions

Main RQ

How is the concept of security testing being used in business contexts?

Attempting to collect valuable data to answer our macro question formally may be something generally complex and even reductive for a concept as complex as security testing. However, considering the general trend of data analysis—as described in the previous sections—it is certainly easy to see that security testing, as established by the research, is a highly variable and evolving concept.

However, we can say that most companies conduct security tests on developed

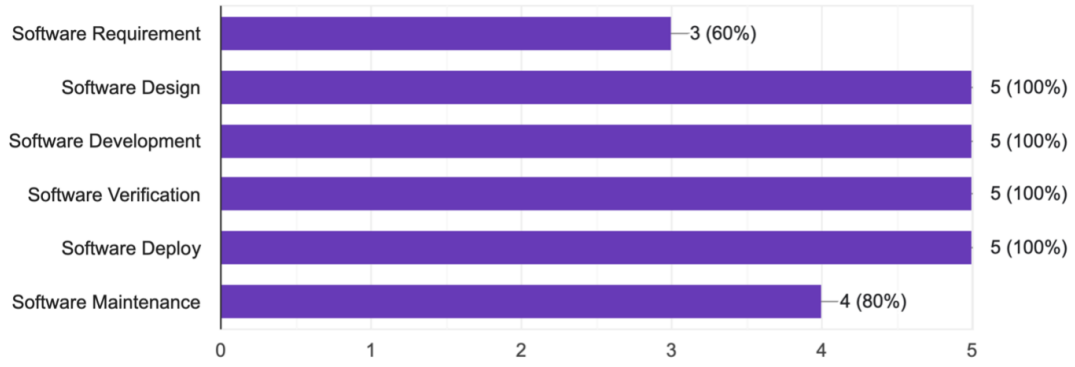

Figure 5.14: Security Testing Activities in Software Development Life-cycle for SAP

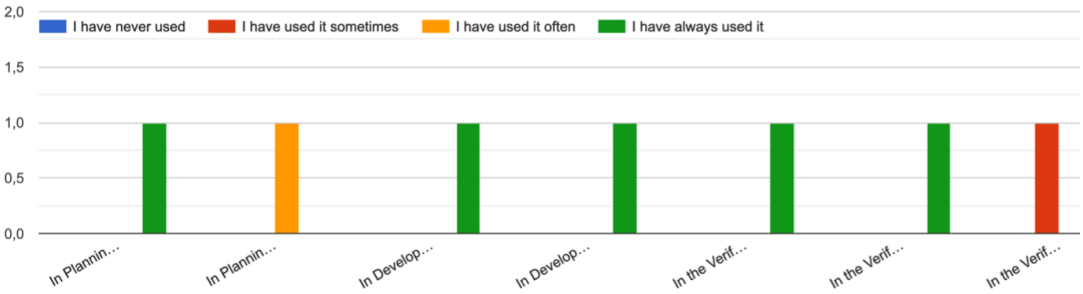

Figure 5.15: Security Testing Technique For Each Phase

software. What does change, however, are the internal organizations; Zucchetti prefers to follow the OWASP framework but builds an internal organization such as ethical hackers to minimize the risk of releasing insecure software. In addition to using a secure lifecycle, development must also be managed from a security perspective, using DevOps and DevSecOps. SAP uses the secure life-cycle along with most of the proposed techniques, however, it believes that each type of application should have its own security techniques. As can be seen, the thoughts are not conflicting; rather, they follow different lines of thought, but the ultimate goal is the same. On the one hand, SAP makes the techniques vary depending on the type of application and depending on the type of architecture. On the other Zucchetti always uses the same testing techniques but all applications to be developed will have a previously tested architecture Last but not least is I.T.Svil, which does not use any framework based on secure life-cycle but applies some security techniques for testing.

RQ_1 - Security Tester Figure

RQ_1

Who might perform security testing in business company?

It is clear from the results that organizations that do not use a specific framework often do not have a specific person in charge of security activities, but that they are performed by the same people who develop the application. In other cases, there are several dedicated security officers who perform the above activities, supported by a strong internal organization and frameworks. Moreover, research has shown that operationalizing employees with security-specific backgrounds can better manage both processes and artifacts of the software security life-cycle compared to companies that delegate the execution of security activities to regular developers. Dr.Piccoli says: "***Frameworks must be used by people who have the right skills, otherwise developers may not consciously apply them.***"

RQ_1: summary of the results. Our results note that most companies have professionals who perform security testing activities. In this way, all activities and processes related to frameworks take on a strong value.

RQ_2 - Who Is Responsible for the Application of Security Frameworks??

RQ_2

Are we sure that all IT companies use framework based on secure development life-cycle?

As can be seen from the data analysis in the previous section, not all organizations choose to use a life-cycle-based security framework to conduct security activities. We concluded that this may depend on a number of factors:

- A lower level of process organization than other companies.
- The size of the company in terms of its willingness to hire people with a different background than programmer

- The type of projects it develops

One possible solution could be to delegate security activities to external companies, but this would jeopardize the project delivery schedule. Dr. Piccoli says: "***We take special care to manage all processes, especially those related to security, to the best of our ability. Our teams, both development and security, are constantly expanding.***"

RQ_2: summary of the results. The use of a secure development life-cycle is strictly dependent on certain business characteristics, e.g., company dimension. Some companies prefer to collaborate with external companies to solve security aspects.

RQ_3 - Companies' Security Testing Organization

RQ_3
How are security testing procedures organized during software development?

Organization processes for security testing activities are closely related to business decisions. However, it would be useful to establish process standards for security testing. Every company should follow a security framework based on the secure life-cycle, establishing early in the development phase, the possible security and privacy criticality of the system. It would be useful to have a team of people with a prevalent background in security applies the testing techniques for each phase of the software life cycle. The same team should communicate with the development team to resolve possible issues. Finally, as an additional analysis, one could delegate to outside companies—such as OWASP Italy or Minded Security—further security testing. Dr. Piccoli says: "***Using frameworks, we can manage each stage of software development through a group of ethical hackers, whose purpose is to detect possible security mistakes made by developers.***"

RQ$_3$: summary of the results. Most companies confirm that, the position of Security tester enables continuous improvement of business processes, especially if these activities are carried out by groups of people.

RQ$_4$ - Teamwork

RQ$_4$
Who is the Security Testing process coordinated by? How often is security testing conducted in groups?

The organization chart of the teams should not deviate much from the current assignments. Companies should hire teams of people tasked with testing software security. The Figure 5.16 shows that the team should be autonomous and different from the development team, but above all, it should work as a team to highlight as many critical issues as possible, giving the developers a chance to be able to solve the problems. However, there should be identified–for each development team–a person in charge of managing the processes and the whole communication between the security team and the development team, in order to decrease the skills gap. "***Our ethical hacking team continuously communicates with the development team and its internal security manager. Thus, each phase of the software life-cycle is managed using the frameworks.***"

RQ$_4$: summary of the results. Working in teams allows the developers and security tester teams to stay in communication at all times, although a managerial figure is needed to manage them

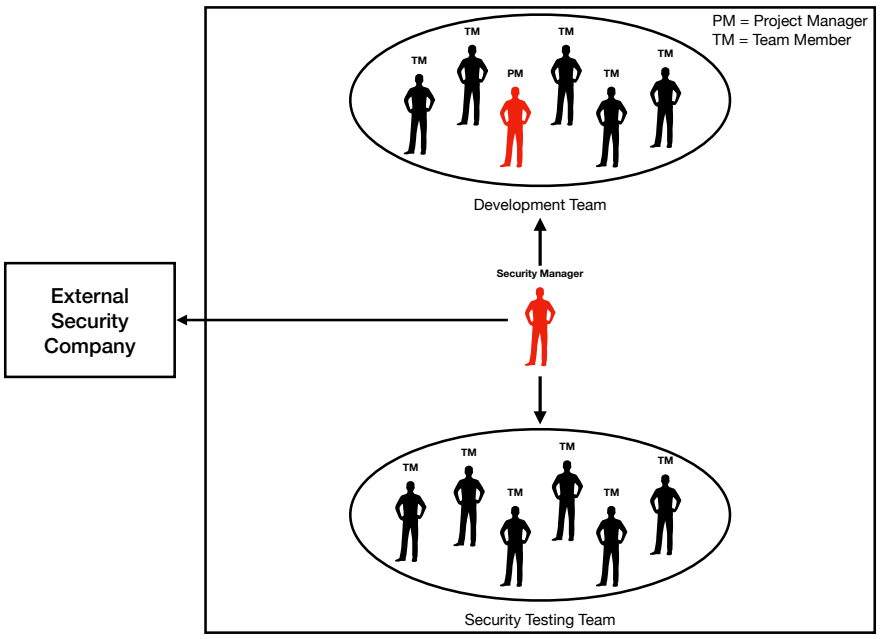

Figure 5.16: Security Testing Team Composition

CHAPTER 6

Threat To Validity

Several factors could affect the validity of the empirical study, and the threats may vary and affect different aspects of the study. To this end, the main strategies to deal with the aspects of empirical validity of the study are considered below, summarizing them according to the four categories defined by Wohlin et al. [31] for software engineering studies: *Construct* validity, *Internal* validity, *Conclusion* validity, and *External* validity.

6.1 Construct Validity

Threats to *construct* validity concern with the relation between theory and observation and are mainly due to imprecision in performed measurements [31].

The research methodology used is the Survey dedicated to IT companies, for:

- Obtain information from various professionals (Software Engineer, Security Specialist, Project Manager, CEO, CTO, etc).
- Obtain homogeneous and targeted data that can be easily analyzed

In designing the survey, the relationship between hypothesis and results was taken into account; in fact, it is possible to relate the survey questions and the research

objectives. As can be seen from the summary tables 4.3 4.3, the questions were asked in English to account for the possible cultural differences of the participants. The vocabulary used is simple and fluent and avoids overly long or complicated sections. One of the biggest risks is creating a lengthy survey, as this can easily lead to a lapse in attention, making the responses collected invalid. One possible plan to avoid this is to conduct a pilot test, in order to also mitigate construct validity. The test was conducted with master's students from the University of Salerno and with young developers from different companies. Both were attending a degree program at the University of Salerno. The participants were asked to fill in a copy of the electronic survey to estimate the running time and to identify the presence of unclear questions or definitions. Participate requirements for the pilot test are:

- Know at least one high-level programming language.
- Have participated in a course or work in the security field.
- General knowledge of software engineering

All answers from the pilot test are not considered in the evaluation of the results. The results of the pilot test were satisfactory: the 5 participants—including one with dyslexia—completed the survey within the allotted time. The participant with dyslexia completed the survey, taking longer but staying within the allotted limits. Participant feedback was consistently positive, with the exception of some questions that were later reworded.Another, no less important aspect is to ensure the repeatability of the study. In addition, all documents, resources and the dissertation itself will be available on the GitHub repository of the research lab SeSa Lab (Software Engineering Salerno).[1]

6.2 Internal Validity

Threats to *internal* validity concern factors that might have influenced the obtained results without the researcher's knowledge [31].

[1] `https://github.com/sesalab`

With regard to possible threats due to lack of experience in the specific area requested, it is necessary to consider the various ways of disseminating the survey and the subsequent cleaning operations before starting the actual procedure of analyzing the collected responses. The dissemination of the survey was done through direct acquaintances with the human resources department of IT.Svil and SAP. Zucchetti, on the other hand, was contacted through a special email provided on the official website.[2] Given the possible threats and the possible presence of sample bias, a number of Data Cleaning strategies were designed to transform the previous source results into a real analyzable data set. The criteria chosen have already been exposed at the end of the Section 4.2.2. Given the limited results obtained, the use of a single interview with a single company is recommended. Subsequent interviews can then be the subject of a future study to obtain developmental comparisons.

6.3 Conclusion Validity

Threats to *conclusion* validity concern the relation between treatment and outcome and are related to issues that affect the ability to draw the correct conclusions at the end of the work [31].

In relation to the research objectives set, the type of study conducted observed how descriptive statistics instruments can be adapted to specific needs compared to other analytical methods. The specific survey questions are mainly of two types:

- Structural, multiple-choice questions with single or multiple values (multiple responses for a single question)
- Open-ended questions are used mostly for business and personal information.

Another important decision related to the validity of the conclusions is to define the specific analytical tools for each research objective: As will be seen in the next chapter on analysis, several graphical tools were considered for each observation that would best allow aggregating the data and describing the results obtained, especially among the different descriptive statistics tools available—i.e., pie chart and bar diagram—, which were chosen as needed.

[2] `https://www.zucchetti.it/website/cms/contatti.html`

6.4 External Validity

Threats to *external* validity are conditions that limit our ability to generalize the results of our experiment to the real world [31].

To conclude finally, regarding the overview of threats to the validity of the study, it is certainly appropriate to place some considerations about external validity, which effectively encompasses what is the level of generalization of the results. The study already aims to learn about current security practices within companies, however, all companies considered operate in Italy and all participants are Italian. For future developments, it might be useful to extend the work to companies outside Europe in order to have more external validity to the results.

CHAPTER 7

Conclusion and Future Work

This chapter illustrates a summary of the research and future prospects.

7.1 Conclusions

The main objective of this study is to shed light on security testing practices performed by practitioners in the context of software development. Specifically, our intention was to:

- Determine whether companies have employees performing security testing activities to release reliable software.
- Determine whether companies use frameworks based on the secure Development Life-cycle.
- Understand what techniques companies prefer to use and especially how security testing procedures are organized from the project's beginning to release.
- Understand whether there is a figure inside or outside the teams who coordinates security work.

In order to reach our objectives, we adopted a qualitative research approach, questioning participants from three IT companies—i.e., Zucchetti S.p.A, SAP S.p.A, and I.T.Svil S.r.l. Specifically, we performed A survey designed to find out whether participants have specific skills in development, testing, and cryptography. In addition, we asked whether they used security frameworks and what security testing techniques they used. Furthermore, we performed an interview with the CTO of Zucchetti S.p.A to obtain additional details left out in the survey.

We found that not all companies use security frameworks, and it follows that some testing techniques lack proper organization. Moreover, two of the three companies surveyed say they have a figure with the right skills and that the entire process is almost always done in groups of people, as a team.

To sum up, a security tester must have specific skills in security and cryptography other than in software development—to ensure proper communication with the development team—and software testing. Furthermore, software development teams should use frameworks based on secure development during the software life cycle. As a final contribution, we provide all the data and tools used during our research in an online appendix[1] in order to encourage study replication.

7.2 Future Work

In this section, we describe our future research agenda.

Identification of vulnerabilities An initial future development regards the identification of security testing techniques based on the type of application to be developed and the software architecture changes. This contribution would be helpful in providing further guidelines for the creation of a secure development standard.

Tool Development A second development concerns the creation of a tool for source code analysis, providing a prediction of potentially dangerous code. The combination of the two works could fully support developers in producing and analyzing source

[1] `https://bit.ly/3QUDSo0`

code from a security perspective. In addition, the tool will be able to suggest—depending on the architecture and modulation of the code—the best security testing techniques to adopt.

Quantitative Studies. Last, in the future, we plan to perform quantitative studies—e.g., data mining and statistic analysis—in a *mixed-method research* fashion [32, 33, 34]. We aim to complement the results obtained in order to (1) validate the qualitative study finding and (2) strengthen their generalizability.

Bibliography

[1] Nor Hassan et al. "Enhancing the Secured Software Framework using Vulnerability Patterns and Flow Diagrams". In: *International Journal of Advanced Computer Science and Applications* 9 (Jan. 2018). DOI: 10.14569/IJACSA.2018.090946.

[2] Chris Wysopal et al. *The Art of Software Security Testing: Identifying Software Security Flaws (Symantec Press)*. Addison-Wesley Professional, 2006. ISBN: 0321304861.

[3] Ian Sommerville. *Software Engineering*. 10th. Pearson, 2015. ISBN: 0133943038.

[4] Allen H. Dutoit Bernd Brugge. *Object-Oriented Software Engineering Using UML, Patterns, and Java*. 2nd. 2004, pp. 25–100.

[5] E.J. Byrne. "A conceptual foundation for software re-engineering". In: *Proceedings Conference on Software Maintenance 1992*. 1992, pp. 226–235. DOI: 10.1109/ICSM.1992.242539.

[6] "ISO/IEC/IEEE International Standard - Systems and software engineering – Software life cycle processes". In: *ISO/IEC/IEEE 12207:2017(E) First edition 2017-11* (2017), pp. 1–157. DOI: 10.1109/IEEESTD.2017.8100771.

[7] L. A. Belady and M. M. Lehman. "A model of large program development". In: *IBM Systems Journal* 15.3 (1976), pp. 225–252. DOI: 10.1147/sj.153.0225.

[8] Nayan B. Ruparelia. "Software Development Lifecycle Models". In: *SIGSOFT Softw. Eng. Notes* 35.3 (2010), 8–13. ISSN: 0163-5948. DOI: 10.1145/1764810.1764814. URL: https://doi.org/10.1145/1764810.1764814.

[9] Martin Fowler, Jim Highsmith, et al. "The agile manifesto". In: *Software development* 9.8 (2001), pp. 28–35.

[10] B.W. Boehm. "Verifying and Validating Software Requirements and Design Specifications". In: *IEEE Software* 1.1 (1984), pp. 75–88. DOI: 10.1109/MS.1984.233702.

[11] Michael E. Fagan. "Design and Code Inspections to Reduce Errors in Program Development". In: *IBM Syst. J.* 38 (1999), pp. 258–287.

[12] J. M. Juran, F. M. Gryna, and McGraw-Hill Companies. *Juran's Quality Control Handbook*. McGraw-Hill, 1988. ISBN: 9780070331761. URL: https://books.google.by/books?id=_-VTAAAAMAAJ.

[13] Glenford J. Myers, Corey Sandler, and Tom Badgett. *The Art of Software Testing*. 3rd. Wiley Publishing, 2011. ISBN: 1118031962.

[14] Pierre Bourque and Richard E. Fairley, eds. *SWEBOK: Guide to the Software Engineering Body of Knowledge*. Version 3.0. Los Alamitos, CA: IEEE Computer Society, 2014. ISBN: 978-0-7695-5166-1. URL: http://www.swebok.org/.

[15] Ivan Krsul. "Software Vulnerability Analysis". In: *ETD Collection for Purdue University* (Jan. 2011).

[16] B. Potter and G. McGraw. "Software security testing". In: *IEEE Security Privacy* 2.5 (2004), pp. 81–85. DOI: 10.1109/MSP.2004.84.

[17] Shanai Ardi et al. "How can the developer benefit from security modeling?" In: *The Second International Conference on Availability, Reliability and Security (ARES'07)*. 2007, pp. 1017–1025. DOI: 10.1109/ARES.2007.96.

[18] G. McGraw. "Software assurance for security". In: *Computer* 32.4 (1999), pp. 103–105. DOI: 10.1109/2.755011.

[19] In.

[20] S. Barnum and G. McGraw. "Knowledge for software security". In: *IEEE Security Privacy* 3.2 (2005), pp. 74–78. DOI: 10.1109/MSP.2005.45.

[21] Matthew Nicolas Kreeger. "Security Testing: Mind the Knowledge Gap". In: *SIGCSE Bull.* 41.2 (2009), 99–102. ISSN: 0097-8418. DOI: 10.1145/1595453.1595484. URL: https://doi.org/10.1145/1595453.1595484.

[22] H.H. Thompson. "Why security testing is hard". In: *IEEE Security Privacy* 1.4 (2003), pp. 83–86. DOI: 10.1109/MSECP.2003.1219078.

[23] Hari Maurya. *Microsoft Security Development Lifecycle (SDL)*. 2010. URL: http://techsurface.com/2010/01/microsoft-security-development-lifecycle-sdl.html.

[24] .

[25] Pravir Chandra. *Software Assurance Maturity Model*. URL: https://opensamm.org/downloads/SAMM-1.0.pdf.

[26] Mark C Paulk et al. "Capability maturity model, version 1.1". In: *IEEE software* 10.4 (1993), pp. 18–27.

[27] Mehmet Kara. "Review on common criteria as a secure software development model". In: *International Journal of Computer Science & Information Technology* 4.2 (2012), p. 83.

[28] Michael Howard and Steve Lipner. *The Security Development Lifecycle. Secure Software Deelopment, Redmond*. 2006.

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

[30] Sue Wilkinson. "Focus group methodology: a review". In: *International Journal of Social Research Methodology* 1.3 (1998), pp. 181–203. DOI: 10.1080/13645579.1998.10846874. eprint: https://doi.org/10.1080/13645579.1998.10846874. URL: https://doi.org/10.1080/13645579.1998.10846874.

[31] Claes Wohlin, Martin Höst, and Kennet Henningsson. "Empirical research methods in software engineering". In: *Empirical methods and studies in software engineering*. Springer, 2003, pp. 7–23.

[32] John W Creswell and J David Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

[33] Massimiliano Di Penta and Damian Andrew Tamburri. "Combining quantitative and qualitative studies in empirical software engineering research". In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE. 2017, pp. 499–500.

[34] R Burke Johnson and Anthony J Onwuegbuzie. "Mixed methods research: A research paradigm whose time has come". In: *Educational researcher* 33.7 (2004), pp. 14–26.

CHAPTER 8

Ringraziamenti

In primissimo luogo vorrei ringraziare il Prof. Palomba, Stefano, Valeria ed Emanuele per aver seguito passo dopo passo l'intero lavoro di tesi. Ringrazio inoltre tutti gli altri componenti del SeSa Lab: Dario, Manuel, Giammaria, Francesco e Giulia per avermi accolto all'interno del laboratorio. Grazie ragazzi!

Vorrei ringraziare le aziende: **I.T.Svil**, **SAP** e **Zucchetti** ed ai professionisti che hanno partecipato al Survey. Grazie a voi ho avuto l'opportunità di contribuire per la prima volta al mondo della ricerca, occupandomi di una nuova delicata tematica. Nello specifico, ringrazio Antonino Sabetta, principal research scientist di SAP e la dottoressa Anna Troisi, HR manager di I.T.Svil per aver gentilmente concesso ai professionisti delle rispettive aziende di partecipare al questionario.
Un ringraziamento speciale va a **Gregorio Piccoli**, membro del consiglio di amministrazione e CTO di Zucchetti, che, sin da subito con grande disponibilità mi ha riportato la sua esperienza e quella dell'azienda in ambito Security Testing. Fonte di ispirazione, estremamente qualificata con cui spero, in futuro, di poter collaborare.