End-of-Master-Project

S-SDLC IMPLEMENTATION IN A GAMIFIED E-LEARNING PLATFORM

September 2021

UNIVERSIDAD DE ALCALÁ

Author: Ismael Jiménez Castro

Tutor: Susel Fernández Melián

Co-tutor: Javier Junquera Sánchez

Universidad de Alcalá

Máster Universitario en Ciberseguridad

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Máster Universitario en Ciberseguridad

End of Master Project

S-SDLC Implementation in a gamified e-learning platform

Author: Ismael Jiménez Castro

Tutor: Susel Fernández Melián

Co-Tutor: Javier Junquera Sánchez

Tribunal:

Presidente: Miguel Ángel Sicilia Urbán

Vocal 1º: Iván Marsá Maestre

Vocal 2º: Manuel Sánchez Rubio

Calificación: .....

Fecha: .....

The beginning of the long road ahead...

"He who thinks great thoughts, often makes great errors"

Martin Heidegger

Acknowledgements

I have always pursued to find my passion, something that would make me enjoy to do what I do. This masters just constitutes to the first step of many. It has not been easy to get to this point, yet here I am. This, however, is something I could not have done this on my own.

I want to say thank you to my family, girlfriend and relatives, who have always been there to support me. Special honours to the professors who have helped me along this on-year-beautiful trip, particularly Susel Fernández Melián and Javier Junquera Sánchez for giving me the assistance and help to carry out this project. I am very grateful to count on wonderful people and I cannot think of a better way to thank them but by dedicating them this project. My achievements are also theirs.

Resumen

Este proyecto de fin máster está destinado a la creación de un ciclo de vida de desarrollo seguro del software a partir de una aplicación web que ya ha sido desarrollada. De esta forma, se dejan atrás las metodologías tradicionales del diseño y desarrollo de software en las que la seguridad no es importante, consideradas inseguras y poco eficientes. En su lugar se crea un S-SDLC destinado a aplicaciones web apostando siempre por la seguridad en el software, pero siempre proporcionando el servicio para el que está destinado.

El ciclo de vida de desarrollo de software seguro diseñado en este proyecto está inspirado en el MS-SDL, incluyendo las distintas fases que lo componen e conteniendo algunas modificaciones para hacer de este S-SDLC desarrollado uno completo, seguro y robusto.

En el proyecto se ordena en diferentes fases en las que se aporta un breve introducción, seguida de un marco teórico en el que se cubren conceptos clave, destacando el funcionamiento de las diferentes fases del S-SDLC de Microsoft y las diferentes secciones que componen un SDLC. Posteriormente, se definirá un S-SDLC para aplicaciones web inspirado en el MS-SDL y se aplicará sobre la aplicación web en cuestión para analizar sus resultados posteriormente y sacar las respectivas conclusiones, planteando trabajos futuros.

Palabras clave: S-SDLC, SDLC, Seguridad, Software, Aplicación Web

Abstract

This end of master's project is aimed at the creation of a secure software development life cycle from a web application that has already been developed. In this way, we leave behind the traditional methodologies of software design and development in which security is not important, considered insecure and inefficient. Instead, an S-SDLC is created for web applications, always betting on software security, but always providing the service for which it is intended.

The secure software development life cycle designed in this project is inspired by the MS-SDL, including the different phases that compose it and containing some modifications to make this S-SDLC developed a complete, secure and consistent one.

The project is organized in different phases in which a brief introduction is provided, followed by a theoretical framework in which key concepts are covered, highlighting the operation of the different phases of Microsoft's S-SDLC and the different sections that make up an SDLC. Subsequently, an S-SDLC for web pages inspired by the MS-SDL is defined and applied on the web page in question in order to analyze its results later and draw the respective conclusions proposing future work.

Key Words: S-SDLC, SDLC, Security, Software, Web Application

Contents

Acknowledgements **v**

Resumen **vii**

Abstract **ix**

List of Figures **xvii**

List of Tables **xix**

1 Introduction **1**

1.1 Introduction to the Project 1

1.2 Project rationale 2

1.3 Objectives 3

1.4 Organization of the Project's Memory 4

2 Theoretical Background **7**

2.1 State of Art 7

2.1.1 Software 7

2.1.2 Security 8

2.1.3 Web Application 9

2.1.4 Web Application Flow 9

2.1.5 Benefits of web apps 10

2.1.6 SDLC 11

2.1.7 S-SDLC 11
2.2 Study of the Environment. A Deeper Analysis of Web Security, SDLC & S-SDLC 12
2.2.1 Security on web Applications 12
2.2.1.1 OWASP TOP 10 14
2.2.1.2 Security Principles in Web applications 15
2.2.2 SDLC: a deeper analysis 17
2.2.2.1 Traditional Methodologies 18
2.2.2.2 Modern Methodologies: Agile 22
2.2.3 The Secure SDLC 23
2.3 Definition of S-SDLC Stages 26
2.3.1 Preparation: Requirements gathering 27
2.3.1.1 Training must be Provided 27
2.3.1.2 Definition of Security Requirements 28
2.3.1.3 Metrics Definition and Compliance Reporting 28
2.3.2 Design: Threat Modeling and Design Review 29
2.3.2.1 Threat Modeling Performance 29
2.3.2.2 Design Requirements Establishment 29
2.3.2.3 Definition and Usage of Cryptography Standards 30
2.3.2.4 Management the Security Risk of Third-Party Components 30
2.3.2.5 Definition of Approved Tools 30
2.3.3 Development: Static Analysis 30

2.3.3.1 Performance of a Static Analysis Security Testing (SAST) 31
2.3.4 Testing: Security Testing and Code Review 31
2.3.4.1 Performance of a Dynamic Analysis Security Testing (DAST) 32
2.3.4.2 Performance of a Penetration Testing 32
2.3.5 Deployment and Maintenance: Security Assessment and Security Configuration 33
2.3.5.1 Establishment of a Standard Incident Response Process 33

3 S-SDLC for Web Applications 35

3.1 First Stage: Preparation 35
3.1.1 Web Applications' Training 35
3.1.2 Security Requirements of the Web Application 40
3.1.3 Metrics definition and Compliance Reporting for the Web Application 41
3.2 Second Stage: Design 43
3.2.1 Threat Modeling Performance 43
3.2.2 Design Requirements Establishment 46
3.2.3 Definition and Usage of Cryptography Standards . . . 47
3.2.4 Management the Security Risk of Third-Party Components 48
3.2.5 Definition of Approved Tools 49
3.3 Third Stage: Development 53

3.3.1 Performance of a Static Application Security Testing (SAST) 54
3.4 Fourth Stage: Testing 54
3.4.1 Performance of a Dynamic Application Security Testing (DAST) 55
3.4.2 Performance of a Penetration Testing 56
3.5 Fifth Stage: Deployment and Maintenance 57
3.5.1 Safe Deployment Requirements 58
3.5.2 Establishment of a Standard Incident Response Process 58

4 S-SDLC Implementation in the E-Gamified Platform 61

4.1 Preparation Implementation 61
4.1.1 Training and security compliance 61
4.1.2 Security Requirements of the Web Application 67
4.1.3 Metrics definition and Compliance Reporting 71
4.2 Design Implementation 72
4.2.1 Threat Modeling Performance 72
4.2.1.1 Threat Model Approach 73
4.2.1.2 Application's Architecture and Use Cases . . 73
4.2.1.3 STRIDE strategy to finding threats 75
4.2.1.4 Risk management 78
4.2.2 Design Requirements Establishment 83
4.2.3 Definition and Usage of Cryptography Standards . . . 83
4.2.3.1 Django as a Web application framework . . . 83

4.2.3.2 Database data encryption 83
4.2.3.3 Encryption within the transportation of information 84
4.2.4 Management the Security Risk of Third-Party Components 84
4.2.5 Definition of usage of approved tools 85
4.3 Developing 86
4.4 Testing: Security Testing and Code Review 88
4.4.1 Dynamic Testing 88
4.4.2 Penetration Testing 89
4.5 Deployment and Maintenance 89
4.5.1 Safe Deployment Requirements 89
4.5.2 Establishment of a Standard Incident Response Process 91
4.5.2.1 Incident Response Plan 92
4.5.2.2 Business Continuity Plan 94

5 Results **97**

5.1 Preparation results 97
5.2 Design results 98
5.3 Development results 101
5.3.1 Password enforcement policies 101
5.3.2 Implementation of a two factor authentication system (admin) 104
5.3.3 Static analysis 105
5.4 Testing results 110

5.4.1 Results of the dynamic analysis 110
5.4.2 Penetration testing evidences 114
5.4.3 Burp Suite Brute Force attacks 114
5.4.4 Other tools usage 116
5.5 Deployment and maintenance results 117
5.6 Closing Lines 118

6 Conclusions **121**

Bibliography **123**

List of Figures

2.1 Waterfall Model. Source: i1.wp.com 18

2.2 V-Shapped Model. Source: slidebazaar.com 19

2.3 Iterative Model. Source: i0.wp.com 20

2.4 Iterative Model. Source: Eternal Sunshine is the Mind 21

2.5 Big Bang Model. Source: pngitem.com 22

2.6 Agile Model. Source: rnftechnologies.com 23

2.7 Cost to fix a bug by development phase. Source: IBM System Science Institute 24

2.8 S-SDLC. Source: Snyk.com 27

4.1 Web Application System 73

4.2 SonarLint Plugin functioning 86

4.3 SonarLint Extensive explanation 87

4.4 OWASP-ZAP Analysis 88

5.1 User's hashed data 99

5.2 OWASP Dependency Check Running 100

5.3 OWASP Dependency Check Report 100

5.4 Email checking 102

5.5 Password checking 103

5.6 Results in the web application 103

5.7 QRcode generation 104

5.8 Token Required to log in 105
5.9 Token Required generated in the phone 105
5.10 SonarQube Login 106
5.11 SonarScanner Configuration 106
5.12 Several Bugs 107
5.13 Security Hotspots 107
5.14 Results in SonarQube 108
5.15 New Analysis 108
5.16 CSRF possible attack 109
5.17 CSRF possible attack solution 109
5.18 Analysis to the quiz application 110
5.19 Cross-Domain JavaScript Source File Inclusion 111
5.20 Cross-Domain JavaScript Source File Inclusion 111
5.21 X-Content-Type-Options Header Missing 112
5.22 Solution to X-Content-Type-Options Header Missing 113
5.23 Suspicious comments that must be avoided 113
5.24 Caught introducing text 114
5.25 List of users 115
5.26 Found name and password 116
5.27 WAF detection 117

List of Tables

4.1 STRIDE Properties Table 68
4.2 Metrics Table 72
4.3 Spoofing Threats Risks Table 79
4.4 Tampering Threats Risks Table 80
4.5 Repudiation Threats Risks Table 81
4.6 Information Disclosure Threats Risks Table 81
4.7 Denial of Service Threats Risks Table 82
4.8 Elevation of Privilege Threats Risks Table 82

Chapter 1

Introduction

1.1 Introduction to the Project

The advance of technology is unstoppable. It keeps growing day by day, introducing itself into our lives in such ways we would have never imagined. And so does software. Technology cannot now be understood without it, since software has had the greatest impact in the technology evolution process. However, as in many other aspects in life, progress also brings us bad consequences and software is not an exception: cyber risks keep increasing day by day.

Software has been designed and developed during many years following a series of traditional methodologies following what we know now as SDLC (Software Development Lifecycle). These are now obsolete, since they do not put too much care into cyber security. After all, we are nowadays aware that the integration of security is a necessity and new methodologies have appeared to fix these issues in the form of Secure Software Development Lifecycle (S-SDLC).

In this project, a S-SDLC will be designed and implemented from a previously designed application, so that is satisfies the requirements to be considered a secure designed and developed one.

1.2 Project rationale

In the year 2017, an innovative project was launched in Universidad de Alcalá named *"Aplicación de nuevas tecnologías para el apoyo a la docencia en un entorno interactivo"* [1]. Here, advantages and disadvantages of commercial software tools were analyzed to study the interactions between teachers and students and the different solutions where studied in order to improve the existing flaws. A next step was taken with the project *"Nueva Aplicación Multiplataforma para el Aula Invertida. Prueba Piloto y Despliegue"*[2], where the proposal of designing a new application was initiated, pursuing an improvement of the commercial material.

A first application was created in the project *"Web Sockets Implementation in the Application for the Flipped Classroom"* [3]. It consisted on a basic app where the students answered to multiple questions proposed by the teachers, and then the teacher would see the results of such responses. The communication was established through web sockets that made it possible to see actions taken at real time.

As for the service, the software works fine. But, as said, the application has not been developed thinking of a secure environment, where attacks may take place and that we should be aware of. Therefore, this project will pursue to securing a software to securely make use of it by creating a S-SDLC. It makes no sense to possess a software which is not ready to face any kind of incident or is vulnerable to multiple types of threats.

It is obvious many flaws and vulnerabilities can be found, probably enough to write a book about it). The objective of this project is none other than

developing an S-SDLC for web applications, apply it and view the benefits it provides. For this reason, the implementation chapter will be focused one one particular problem: **the possibility of students seeing the answers of others when responding to a particular question displayed by a teacher.** While paying attention to the rest of the aspects, the main focus will be put to secure data flow.

1.3 Objectives

The main goal of this project is to create and deploy a secure software development lifecycle (S-SDLC) from a piece of software that has already been created. It consists on a multiple choice web application implemented through web sockets written in Django.

The renowned S-SDLC will be inspired in the MS SDL model, following the main techniques it includes based on the necessities of the application. The reason why this methodology has been chosen is none other than its good structuring, ease of compression and its completeness with respect to other secure development models. Security and privacy [4] are introduced in every single step of the development process, and this is something that is very much desired in software application where users are involved, as security is fundamental and privacy is a basic right to everybody.

Additionally, the designed S-SDLC will be also be suitable to be used for other pieces of software that are not necessary the software for what it is going to be created. This is a very ambitious goal that will make the result of this project a more versatile and handy one.

1.4 Organization of the Project's Memory

The complete document will be divided into the following chapters:

- **Chapter 1: Introduction**

  This chapter is the one just being covered where the project is presented, justified and the general objectives are displayed.

- **Chapter 2: Theoretical Background**

  In this chapter, the theory baseline the project will be covered. The State of Art and Study of the Environment will be carried out here, the core concepts will be explained.

- **Chapter 3: Definition of the S-SDLC for web applications**

  Here, the different concepts of software development will be covered. This is an important chapter where the different phases of the S-SDLC will be defined and explained. Here, the SDLC is created.

- **Chapter 4: S-SDLC implementation in the E-Gamified Platform**

  The implementation of the previous S-SDLC in the application takes place. It is basically a chapter dedicated to see how the S-SDLC works in a real software.

- **Chapter 5: Results**

  This is the chapter where the results will be exposed and discussed covering every S-SDLC phase.

- **Chapter 6: Conclusion**

  This is a closing chapter where the final statements will be written and the future work to be done in order to improve the project.

Chapter 2

Theoretical Background

This chapter will be dedicated to study the theoretical baseline the project approaches. It is essential to have a clear understanding of certain concepts on which this work is based in order to take advantage of what it tells the reader.

A clear evolution will be appreciated in this chapter as what to complexity of concepts concerns: it will start with some basic ones, as well as generic ones, to then progress into some more complicated ones, explicit to the creation and development of a S-SDLC.

2.1 State of Art

2.1.1 Software

Software is a concept familiar to everybody. Nevertheless, it may not be that easy to explain what it is and this is a very important question to solve, since it is the core of this project.

Software as the interface [5] between machines and humans that allows the latter to make use of them. Without software, the physical materials used to build up machines would be completely useless. It consists on a set of instructions written in code used that tell a given machine how to perform a series of tasks. The code used to develop this instructions continuously varies and there are multiple options where to choose from. Python and Java are

two of the most used coding languages used nowadays.

2.1.2 Security

The word security defines the state of something or someone being free of threat or danger [6]. These are big words that apply to the IT world, to what we know as security for the information security. This term refers to the methods, tools and people involved in the process of defending an organizations digital assets from disruption, exploitation or theft [7]. The IT security can be divided into two parts: the physical and information.

- **Physical security** regards the protection of hardware, software, network information, data and people from intrusions, physical actions or other adverse events that could damage a given organization. It is composed of three parts: access control, surveillance and testing.

- **Information security** (infosec) refers to the protection of processes, tools and policies of digital and non digital assets. It includes

This project will focus in the application of a S-SDLC to a web application (covered in the next section), which is piece of software. Therefore, it is important to highlight the concept of software security.

Software security is the *umbrella term* [8] used to describe how software is engineered so that it is able to to continue function correctly after a malicious attack against it takes place or other risks that try to put it under danger [9]. Software security is nowadays implemented within the Software Development Lifecycle, since security has become something fundamental and software security must not be confused with security in software: it must

combine both security mechanisms (such as access control) and security by design [10].

There are several steps to be taken into account, but these will be covered in the Web applications section.

2.1.3 Web Application

Before heading to the next chapter, there is an additional approach that needs to be talked over: Web Apps Fundamentals.

A web application is fundamentally, computer program that utilizes web browsers and web technology to perform tasks over the Internet [11]. The rapid evolution of technology has led to the creation of this software, that permits users all over the world to interact with one another.

A web application is a client server program, which means that it has a client side and a server side [12]. The first one refers to the user that access to the application, programmed in languages such as Java Script, HTML, etc. The other is the one in charge of handling information: maintain, retrieve ans store data. It is written in languages such as PHP or ASP. In the case of the designed app, the front end (client side) is written in JavaScript, HTML and CSS, whilst the back end (server side) is written in Python (using Django Framework).

2.1.4 Web Application Flow

It is important to have a clear idea of how data flows in a web application.

- A request is sent by the user by means of a browser.
- This request goes from the user (client) to a web server, that manages the information and forwards it to a given application server (server).

- The application server decides what to do what this information: processing data, storing it a database, etc. It then sends the response to the web server.

- The web server hands the data back to the user, that will be able to view the content he pursues.

2.1.5 Benefits of web apps

The emergence of web applications implied a strong impact on how users could interact with a service through the internet. Hence, web apps have brought some benefits users never had before.

- Web apps do not need to be installed in a given device, since they are accessed through a web browser. This leads us to a second advantage.

- There are no compatibility errors, since the device where they run is totally irrelevant. As said, they are accessed through a web browser.

- They update themselves automatically. These kinds of apps run in one or more servers reviewed by the company in charge of providing the service, so the user does not have the responsibility to check on the updates to be carried out.

Web applications must be developed in a structured and secure way. Thus, its development is framed into a S-SDLC [13], where several security features for web applications are satisfied. The SDLC and S-SDLC fundamentals are covered in the following subsections to later jump into a deeper analysis of these topics together with web application security.

2.1.6 SDLC

Software development [14] is the process that involves designing, programming, specifying software, as well as maintain source code, applications, etc. As it may be appreciated, it is a very complex task and therefore, the SDLC (or Software Development Lifecycle) was born.

SDLC is [15] is the process that pursues to produce a good quality software spending the least resources and time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use. It is a way to measure and improve [16] the software development process step by step.

The need to implementing security in this process has led us to the Secure SDLC or S-SDLC.

2.1.7 S-SDLC

In the past, the security related activities took place in the testing stage [17]. This is very inefficient and can lead to the opposite of what is thought for: bugs found too late to be fixed or not even found at all. Therefore, the concept of Secure SDLC appears. The main point of this idea is to apply security activities to every single stage in the SDLC. This is the core concept of the project and it will be discussed deeper later on.

Several models have been proposed, but there are three that stand out from the rest:

- **MS Security Development Lifecycle (MS SDL)** [18] : This may be the most important and complete model, as well as one of the earliest.

It was designed and proposed by Microsoft and its main idea is to follow the classic phases of the SDLC.

It must be remarked that this project's S-SDLC will be inspired in this model, based on the benefits it provides in regards with others: flexibility, versatility, etc.

- **OWASP CLASP**: Designed by OWASP and also based in the MS SDL, it stands for *Comprehensive, Lightweight Application Security Process*. It is focused on setting the roles each worker must carry out inside an organization.

- **NIST 800-64**: Developed by the US Department of Commerce organization NIST, it provides security consideration that are used in the US federal agencies.

2.2 Study of the Environment. A Deeper Analysis of Web Security, SDLC & S-SDLC

In the previous chapter, the basic concepts on which the project is based have been explained. In this one, a deeper analysis will be carried to pursue a better understanding of certain concepts: security in web applications, SDLC and S-SDLC. The last of these three topics will be essential to understand in order to proceed to the definition of the stages of this S-SDLC will be composed of in the next chapter.

2.2.1 Security on web Applications

Web applications are the most common attack vector nowadays. This is something not new to us, but numbers may be.

- Following a report issued by Positive Technologies [19], 44% of all web pages have security issues, 48% additional ones are found vulnerable to unauthorized access, and 17% are vulnerable to exploits from an attacker to take over the control of the device. In fact (and this is something people may not be aware of) a 100% web applications have some sort of vulnerability.

- According to Mozilla Firefox, a 93% of all web pages can be attacked by means of XSS (Cross Site Scripting), no matter what programming language the attack is written in.

- Around 90% of all cyber attacks take place against web applications.

The bad hackers' motivation has changed with the years. Once upon a time, attackers launched attacks pursuing prestige and recognition. Today, attackers' main goal is none other than making money and according to Positive Technologies [20], the principal objective is to take advantage of web application vulnerabilities to obtain an economical prize out of them. There are two principal areas to accomplish this task:

1. By infecting and spreading malware through enterprise networks. This malware spreading would be a ransomware, with which the attacker will ask for a given amount of money.

2. The exploitation of a particular vulnerability, disseminating phishing emails targeting bank employees.

Numbers are shocking, but realistic. Web applications have been proved the number one target objective by hackers. Data loss can result in economi-

cal losses of hundreds of millions of euros. On the other side, the more effort is dedicated to protect some data, the more valuable it becomes. Consequently, web applications deserve special attention.

2.2.1.1 OWASP TOP 10

OWASP (pen Web Application Security Project) is a nonprofit foundation that works to improve the security of software [21]. This organization's aim is to help securing the web. For this purpose, a top 10 list with the most dangerous web vulnerabilities has been developed since 2003 and it is renewed every 4 years. The last officially approved version was released in 2017, but a new one has been drafted this year and it is awaiting for final acknowledgement. This will be the one to be now described [22]:

1. **Broken Access Control** (goes up from 5th position) found in 94% of all tested applications.

2. **Cryptographic Failures** (goes up from 3rd position) often leads to sensitive data exposure or system compromise.

3. **Injection** (goes down from 1st position) found in 94% of all tested applications. The difference with the first mentioned one is that 33 CWEs where mapped into this category have the second most occurrences in applications after broken access control (34CWEs). CSRF is now part of this section.

4. **Insecure Design**, new to this list focus on risks related to design flaws. It basically bets on using a S-SDLC and the shifting left methodology.

5. **Security Misconfiguration** goes up one position, found in 90% of all tested applications. XXE is part of this category.

6. **Vulnerable and Outdated Components** (previously known as Using Components with Known Vulnerabilities) goes up from the 9th position.

7. **Identification and Authentication Failures** (previously known as Broken Authentication) goes down from the second position thanks to the increased availability of standardized frameworks.

8. **Software and Data Integrity Failures** is new in this top 10, regarding lack of attention when verifying integrity the software updates, critical data and CI/CD pipelines. It was previously known as Insecure Deserialization.

9. **Security Logging and Monitoring Failures** (previously known as Insufficient Logging & Monitoring). Failures in this category are usually related to visibility, incident alerting, forensics, etc.

10. **Server-Side Request Forgery** has been recently added after the importance industry professionals have given. However, data shows low incidence at the moment.

2.2.1.2 Security Principles in Web applications

Organizations such as OWASP have defined several general security principles that must be fulfilled when designing a web application. These are gathered in the OWASP Development Guide [23]. They include:

1. Minimise attack surface area, restricting functions to reduce potential vulnerabilities.

2. Establish security defaults, meaning the web application must be secure by default with strong security rules, etc.

3. The Principle of Least Privilege (POLP), meaning that a given user should have the minimum privileges required to perform a specific task.

4. The principle of Defence in depth, meaning many security controls with different approaches must be taken into account to assure a better protection to the application.

5. Make sure the web application fails securely in case an error took place to avoid crashing events.

6. Avoid trusting services such as third-party ones for accessing additional functionality or obtaining additional data from a security perspective.

7. Separation of duties is essential to prevent individuals to perform a fraudulent action within an organization.

8. Avoid Security through obscurity or STO, which involves codes to hide important information and enforcing secrecy as the main security technique [24]. Other techniques should be used.

9. Keep security the simplest possible to avoid having mechanism too complex that may lead to an increase in risk of errors.

10. Fix security issues securely. When an error has been found, developers should look for the root of the cause, repair it and test it, rather than just look for a patch.

All these factors have lead us to have security in mind all along the development of the web application. Web applications are just like another piece of software (accessed through a web browser)that will have to follow an SDLC process when they are developed. As it happens in most of the development processes, web developers do not usually give security the importance it has. This resulted in the implementation of S-SLDC to add a layer of security in every phase of the SDLC. These two concepts (SLDC and S-SLDC) are something fundamental and deserve a deeper analysis. Having a clear understanding of them will help us to have a clear view of the main goal of this project: creating an S-SDLC for a web application.

2.2.2 SDLC: a deeper analysis

SDLC (Software Development Lifecycle) is the process through which a software is created. It is essential to build a compact and robust material following an ordered and structured process. As what to this process concerns, there are five basic phases:

- Requirements gathering.
- Design of the features developers want according to the previously gathered requirements and build an architecture out of it.
- Coding and developing software.
- Testing/Verifying the code developed.
- Release and maintain the software crated after it is deployed. Make sure to make it evolve if needed.

In order to achieve a S-SDLC in a structured way, several methodologies have been designed over a many years. They are subdivided into two groups: the traditional methodologies and modern ones.

2.2.2.1 Traditional Methodologies

- **Waterfall Model**

  This is the oldest and most traditional SDLC existing out of all methodologies. It is a very simple one where phases are developed linearly, completed one after another.

  It is best suited for stable requirements, products where it is clear what is wanted, as it very quick to develop, easy to explain and it is easy to verify errors. As a counterpart, it is very ineffective and difficult to go back after a phase is completed.

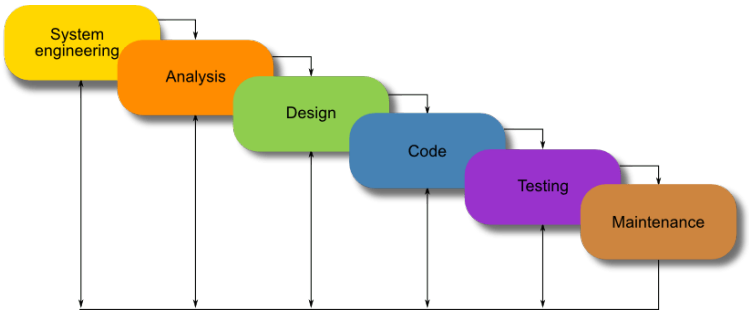

Figure 2.1: Waterfall Model. Source: i1.wp.com

- **V-Shaped Model**

  This methodology is basically an extension to the previously described one, since is has the same base but it introduces a testing phase for each development stage.

  Its benefits are very similar to the waterfall model, but it gives better

results due to the testing phase and ease when talking about process tracking. It is however, hard to go back to make changes if a bug from a previous stage is found.

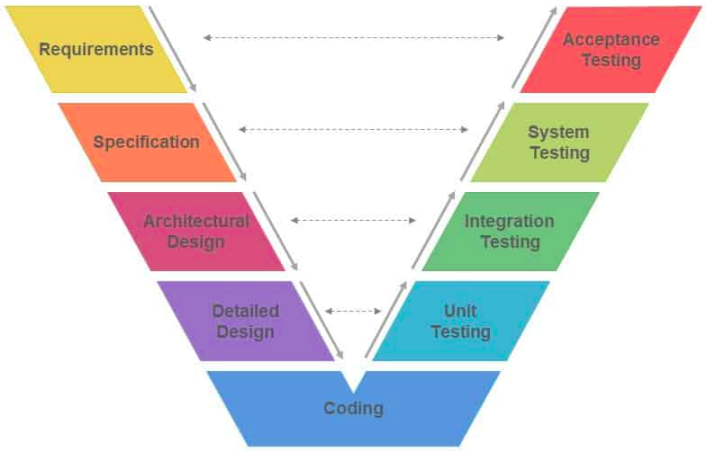

Figure 2.2: V-Shapped Model. Source: slidebazaar.com

- **Iterative Model**

After the project is initialized, it keeps producing new versions of itself after a few requirements are given. The process involves testing, evaluating and allows to dynamically introduce more requirements, while it provides newer results.

It is great when not all requirements are known in the beginning, and it displays a working version early in the process and makes it less expensive to implement changes. Nevertheless, it consumes a lot of resources and this can turn into something inefficient.

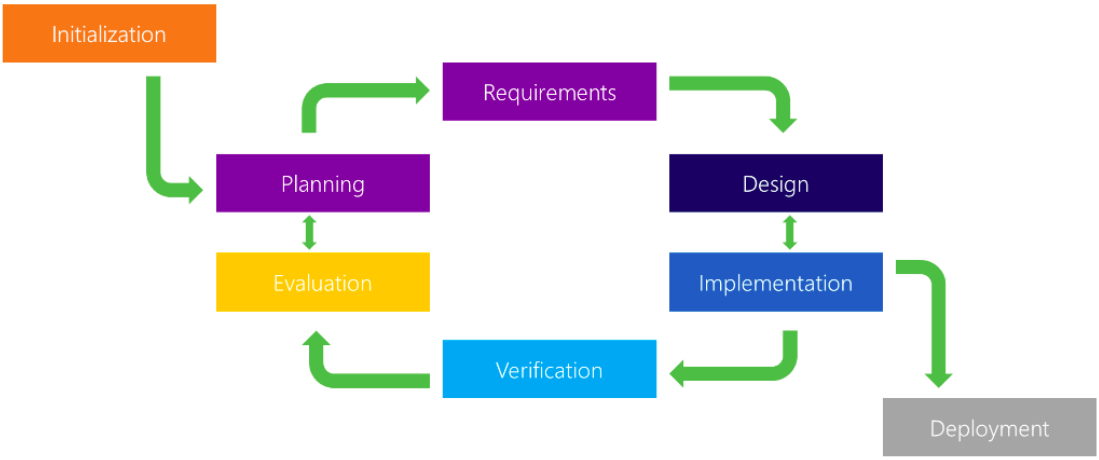

Figure 2.3: Iterative Model. Source: i0.wp.com

- **Spiral Model**

  The spiral model is one of the most flexible methodologies. It is similar to the iterative model, since several phases are repeated in spiral. More concretely, four phases compose this model: planning, risk analysis, design and evaluation.

  The product built using this model will be highly customized and it allows feedback to be incorporated a lot of times since the very early phases of the development. There are however, some counterparts: it can lead to a never-ending system, consuming a lot of human, time and economical resources.

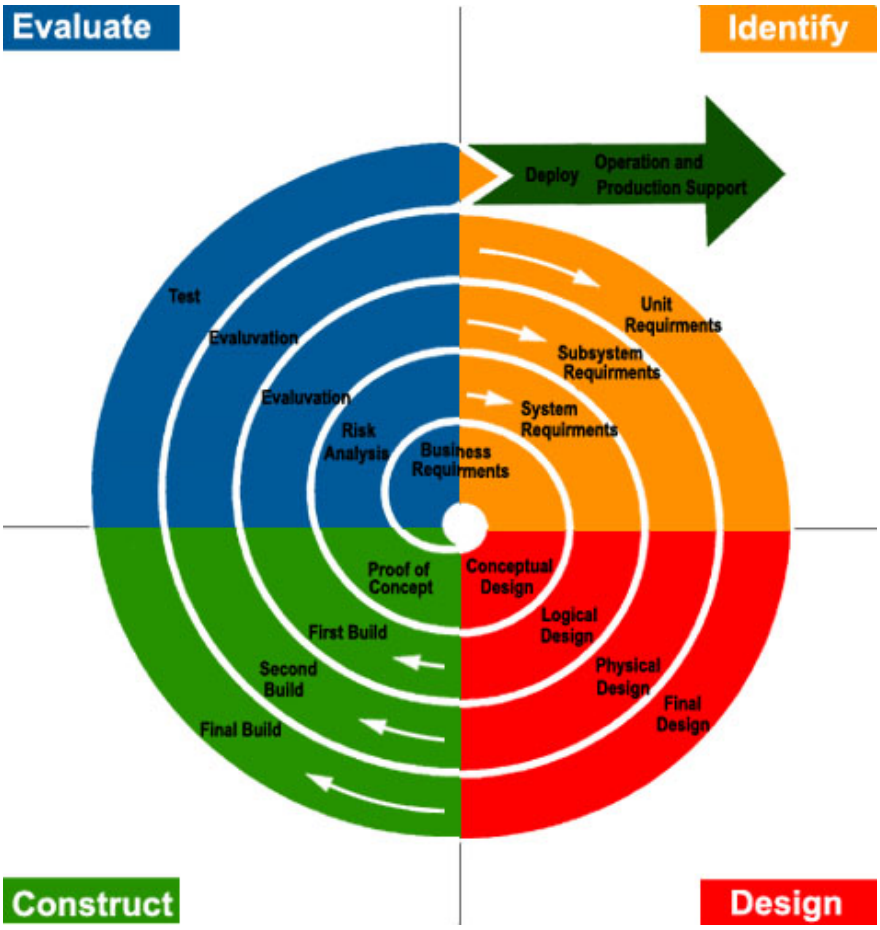

Figure 2.4: Iterative Model. Source: Eternal Sunshine is the Mind

- **Big Bang Model**

  This is a very different approach to software development SDLC to what has been seen until now. The big bang model counts on very little requirements gathering to then build a system: there is very little planning, since all available resources are consumed in the code development phase.

  This is a very unused model because it can end up with a full system that does not count on all requirements the client wants, which turns to be worthless. It may be a good idea for small projects, with a little amount

of members that do not have a high number of requirements (since it is easily manageable and , but unthinkable for bigger-sized ones.

Figure 2.5: Big Bang Model. Source: pngitem.com

2.2.2.2 Modern Methodologies: Agile

- **Agile Model**

  Several SDLC have been covered and yet, yet it is hard to say there is one better than another. This depends on what type of project one is facing. Nevertheless, if that was the question, then the best answer would be the Agile Model.

  This methodology is a combination of the traditional waterfall model and the iterative one, making it flexible. Its main advantage is the continuous evolution of all the phases, from requirements to solutions.

  The model divides tasks into smaller ones at the same time divided into time frames to be practical. The delivering of the results are handed in an iterative way. This allows us to make changes easier and permits

everyone to adapt to each of them dynamically.

This particular of methodology is perfect for projects where the client is very involved in the development process, as well as for projects where the environment changes rapidly.

Among its main strengths, it should be noted quick development, results can be continuously delivered and it is highly adaptive to changes. The number of resources required to carry a given project using Agile is minimum.

Figure 2.6: Agile Model. Source: rnftechnologies.com

2.2.3 The Secure SDLC

So far, Web Applications and SDLC have been covered. Here, security was performed only in the testing part (and not very efficiently). This is quite an inconvenience because of two very important reasons: vulnerabilities are

discovered too late or may not be found at all [17].

Finding vulnerabilities later than expected is a very important problem both functional and economical. According to a study carried out by IBM, fixing a bug in the Testing section is fifteen times more expensive than finding it in the Design one and a hundred times more expensive in the production section. A bar graph is now attached to have a clearer view.

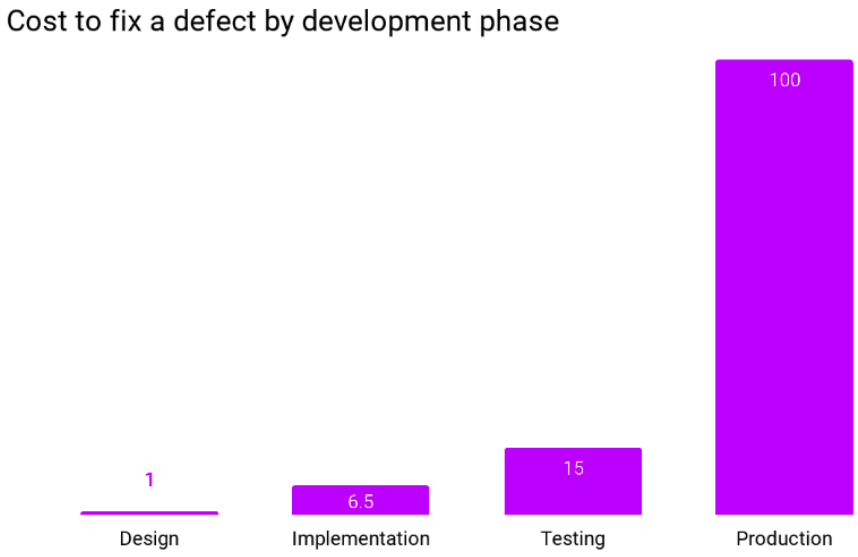

Figure 2.7: Cost to fix a bug by development phase. Source: IBM System Science Institute

This makes it easier to understand how cheaper and fast it is to find mistakes in the development of software earlier in the development process [25]. The idea of securing each of the development phases (introducing this testing section at every layer) led to the creation of the Secure SDLC or S-SDLC.

S-SDLC is a set of all the best practices focused on adding a security layer to the standard phases of the SDLC, from Requirements to Maintenance. Among its main advantages, the most important ones:

- Software is more secure since it a very important concern.
- Clients are aware of the security level of the projects produced.
- Bugs are detected earlier than they where in the past, even earlier than in the coding section.
- Costs are reduced, since the security flaws are found when it is cheaper to fix them.
- The intrinsic business risks of the given organization are reduced.

After this explanation, it is clear what Secure SDLC is and the advantages it provides us. However, there are bunch of best practices everyone involved in the S-SDLC must have in mind:

1. **Developers must be educated**

   In the creation of an S-SDLC there must be something clear: security is everyone's job. This means, every single person involved in the creation of an Secure SDLC must have an understanding of basic security issues or building security into a given software. Training is essential to accomplish this task.

2. **Requirements should be clear**

   No matter what is what one is creating, it must be easy to understand for the developers. They will have to follow some security guidelines, pieces of advice or recommendations and the clearer they are, the easier people will bring solutions to problems and not the opposite.

3. **Be open-minded**

   S-SDLC will alter the way people do their job pursuing more security. They must be ready to interact in order to make changes.

4. **Deal with most important problem first** It is important to face the problems with a higher importance rather than the most urgent ones. Dedicating time for time efficiently to crucial tasks must always prevail over smaller ones that may be easier to do.

It is now the moment to proceed to study the different phases that this S-SDLC will contain adapted to this web application.

2.3 Definition of S-SDLC Stages

Wrapping it up in one sentence, the Secure SDLC consists on the addition of a security layer to each of the stages of SDLC. These stages where described earlier in this chapter and will be now displayed together with the S-SDLC phases that will be applied to each of them. It is important to assume every single person involved in the project has a total knowledge of itself.

As said in the previous sections, this S-SDLC used will be an adapted version of the MS SDL (the one designed by Microsoft). The following is a brief summary of each of the phases of the MS SDL, accompanied with a descriptive image.

Figure 2.8: S-SDLC. Source: Snyk.com

2.3.1 Preparation: Requirements gathering

This is the section where the requirements from every new feature from the different sections of the project is collected. This leads to important security actions to take into account.

2.3.1.1 Training must be Provided

Security is today something essential in the IT world. It is also true every project must count on a cybersecurity expert to oversee how security is integrated in the different project's phases. Nevertheless, it should be clear to every member involved in the project that security is everyone's job. Therefore, workers must have a basic formation in cybersecurity. That means following good practices, working in the correct environment, understanding how to securely use the tools they are given or understand possible vulnerabilities the project may suffer.

Of course, they must be trained to correctly (and securely) to securely make use of the respective software that be required in the project.

2.3.1.2 Definition of Security Requirements

It is important in a S-SDLC to consider security and privacy [26] a fundamental point in order to develop a secure application or system (no matter what the methodology used to develop the software is). Security requirements must be continually updated to to changes in the project's environment or functionalities. Since the methodology to be applied is the agile one, this requirements can be added continuously, but it is preferable to define them at the initial point of the project.

These requirements must be adapted to certain variables that have a direct impact on the way software must be developed. Some of the variables to be included are legal matters, industry affairs, internet standards, coding best practices, review of previous incidents, known possible threats, etc.

2.3.1.3 Metrics Definition and Compliance Reporting

Though this may not be a stage that would not come to one's mind at a first glance, it is essential to count on the minimum acceptable levels of security quality and achieving every single developing team to meet the defined criteria. This is a key factor to help a team understand risks associated with security issues (including identification and fixture of flaws during development or application of standards).

The different issues are categorized with a tag according to its severity. There is a total number of 4 categories: Critical, Important, Moderate or Low.

2.3.2 Design: Threat Modeling and Design Review

As said in the SDLC section, this is the phase there the requirements are transformed into a plan of what it should look like in the actual software application. The SDLC functional part describes what the requirements should do, while on the other hand, the security side focuses on what should not be done.

2.3.2.1 Threat Modeling Performance

Threat Modeling is known as the structured process [27] through which potential security threats and vulnerabilities are identified, their seriousness is quantified and different techniques are prioritized to mitigate attacks and protect resources. All of this within the IT field.

This process must be used in environments where even a minimum risk can appear. Its versatility is very important, since it can be applied at a component, application or even system level and it gives engineers the idea on what security implications the designs must have.

This will then help teams to identify vulnerabilities, determine threats and the risks they will produce and to later produce an structured, robust and secure system/app/component in the future S-SDLC phases.

2.3.2.2 Design Requirements Establishment

At first, the S-SDLC the security features to be applied. Nevertheless, this could backfire us, since sometimes selecting or implementing too complicated security requirements can result to vulnerabilities. This may be an assumable stage, but it has been added considering the importance of understanding and applying the security requirements more consistently and bewaring the

protection they provide.

2.3.2.3 Definition and Usage of Cryptography Standards

Cryptography is critical to ensure data, protect it from disclosure or alteration and many more. Choosing the wrong cryptography can lead to catastrophic effects to the system and it is important to have a clear understanding and assuring the right ones are selected. A good tip may be using approved encryption libraries that allow to be replaced with another in case it is needed.

2.3.2.4 Management the Security Risk of Third-Party Components

Often times (not to say most of them) engineers tend to make use of third party software to be integrated into the one used in the development of the application. When it comes to the designing time, there is the need to have a perfect knowledge of the inventory of this type of software. It is fundamental to understand the impact that a vulnerability in them would have in the system.

Managing the risks they may have will help to decide whether to use a particular software or another.

2.3.2.5 Definition of Approved Tools

This section is very important, since it will determine the tools that the team will use during the development of the project.

A list with the approved tools will be published to be used.

2.3.3 Development: Static Analysis

After the design, a point is reached where the implementation comes into action. In this section, the most important scope security focuses into is

to making sure the code is well-written (secure). This code must of course, follow security coding guidelines and then it must be double checked these reviews have been follow. To perform such task SAST is used.

2.3.3.1 Performance of a Static Analysis Security Testing (SAST)

Static Analysis Security Testing (or SAST) is a testing methodology whose function is to find security vulnerabilities that make a particular system susceptible to suffer an attack [28]. It is also known as white box testing since the tester has access to the underlying framework, design and implementation and it tests the given application from inside out, representing the developers' approach [29].

This kind of analysis is essential to assure secure coding is being followed. It gives ideas on how to replace not-as-secure code with a better one. The team designing the system must agree how often they want to run a SAST to check the system.

These are important advantages, but there is one that may be particularly interesting to the organization in question: SAST allows finding errors earlier in the SDLC, which is a synonym to less expensive fixing: easier and faster to eliminate them. Truly, the main objective in a S-SDLC static analysis is to find the balance between a higher productivity and a good security coverage.

2.3.4 Testing: Security Testing and Code Review

This phase comes after the development to verify whether if the application has met the original design and requirements designed. It is a perfect place where to automated testing is performed or examine security defences with penetration tests (among others).

2.3.4.1 Performance of a Dynamic Analysis Security Testing (DAST)

The Dynamic Analysis Security Testing (DAST) is the complete different approach to the previous analysis studied. Here, a security tool or solution is used to perform a run-time verification [30] of a fully compiled application to check its functionality after all its components have been integrated and running. DAST is also known as Black Box testing, since the tester has no idea of what is being used in the system: frameworks, technologies, etc. It is tested from outside in, representing the "hacker view" by using a set of pre-built attacks that monitor the application behaviour for memory corruption, user privilege issues, and other critical security problems.

This kind of analysis is used in order to fix the issues that the previous SAST has not found: it finds the security bugs by the end of the SDLC, making it more expensive to fix vulnerabilities. Nevertheless, it finds run-time environment related issues and it is used to scan particular pieces of software (in the case of SAST it was used for any kind of software).

2.3.4.2 Performance of a Penetration Testing

This corresponds to the next level of security verification after the DAST has been carried out. A penetration test is carried out by skilled security professionals simulating real-life hacker actions to add an extra layer of security to the system. The main objective is to discover security flaws resulting from coding errors, system configuration faults, or operational deployment weaknesses, including different types of vulnerabilities (counting 0days).

It is important to know they never substitute a DAST, but accompany it. As said, the latter the the vulnerability is found, the more expensive and difficult it is to fix it.

2.3.5 Deployment and Maintenance: Security Assessment and Security Configuration

Contrary to what is popularly believed, the process does not end when the application is released. As known, new vulnerabilities emerge everyday or even errors with which software developers did not count on may appear not only in the code, but also from one of the services there have from third party actors or from open-source code that are used in order to develop the code of the app.

This vulnerabilities must be patched by a development team involving activities such as rewriting code or performing DAST and penetration testing from time to time to find new flaws that can make the application vulnerable.

2.3.5.1 Establishment of a Standard Incident Response Process

An Incident Response Plan is crucial for the maintenance of the application. New threats emerge everyday and the system must be ready for it: the better the plan is, the better the system will recover from an incident. No matter how bad it is. All possible worst-case scenarios must be taken into account when carrying out this plan, as well as responsibilities each person has, who to contact when something happens and several other procedures to follow.

This plan will determine what measures must be taken when an adverse situation is given, fix how often audits should be done, determine the actions each person involved in the system/application should take according to the role they hold, etc.

Chapter 3

S-SDLC for Web Applications

The previous chapter ended up showing the different stages the Web Application in question will undergo, accompanied by the respective theoretical explanation. It is now time to make use of this S-SDLC adapting it for web applications.

The security phases will be designed for a web application according to the previously described S-SDLC (MS SDL). They will be disclosed following such scheme, conforming the various stages to what would suit better a web application (and not any kind of Software).

3.1 First Stage: Preparation

The first stage consists on three main phases that will imply a whole different set of tasks, including formation of the employers and requirements gathering regarding web developing apps, to defining the metrics to be used.

3.1.1 Web Applications' Training

The main goal of a Secure SDLC for a web application is to provide a security assurance process, reducing the risk of vulnerabilities all along the process of web development rather than waiting until the end [31]. Having this is mind, it seems clear understanding the possible security threats and how to deal with them is fundamental in order to developing a secure software

[32]. Implementing an S-SDLC implicitly requires an investment of time and money in the education of the regarding people involved in the project.

It is true some the formation will not be the same for everyone, since the web application will count on developers, designers, analysts, data managers, or even leaders whose knowledge in the cybersecurity environment may be limited. Therefore, there are some concepts everyone show be aware of before diving into the project:

- **Introduction to Cybersecurity.**

  This is quite easy to find, since it is a trending topic in the IT world. YouTube [33] is an open source website where magnificent courses for beginners can be found, such as CBT Nuggets Series or Simplilearn Series. Additionally, Udemy [34] (a virtual academy) offers free introduction courses, such as Cyber Security Course for Beginners - Level 01.

  These cited sources are all video courses, but users may like books better. If that was the case, then Cybersecurity For Dummies by Joseph Steinberg would suit perfectly.

- **Internalize Security as part of design and development**

  It is more common than one would think to find a project where software developers care about developing, leaving security aside. This, of course, can lead to adverse situations where the system (and thus, organization). In most cases, the system is compromised due to vulnerabilities that emerged because of design flaws that could have been prevented if a secure design had been followed. Hence, it is important for coders to internalize security must be part of the software development process.

To help accomplishing it, there is a book that perfectly fits it: Secure by Design. It can be considered a practical handbook that aims to help developers how to use design to bring security into software development, including patterns, best practices, etc. to be applied in the real world and spot vulnerabilities in code and other weaknesses and how they could be fixed [35].

The concept of S-SDLC must properly be understood if a secure web app is what is pursued to be design. Therefore, the MS SDL and OWASP CLASP (based on the previous one) are the two main secure methodologies one must have read and conceive, as well as internalizing it. This project is particularly focused on MS SDL for web pages, so it must be taken as a must dedicating time to understanding such guide.

- **Principles of Threat Analysis**

  Creating a web application is not an easy task and making it secure is even more difficult [36]. Threat Analysis is a technique that involves every team member in the project to discuss, define and document the security aspects in the web application and it is a crucial task to perform in the development of any kind of software. Therefore, people involved in the development of a web application must have a good formation on threat Modeling, to assume why is it useful and how to carry it out.

  To accomplish this task, one must refer to a book called Threat Modeling: Designing for Security that easily explains the basics, why it is needed and different models (particularly S.T.R.I.D.E.), teaching the reader how to manage threats and it even suggests tools to be used

for practical cases [37]. It is very easy to read and a must in S-SDLC development

- **Formation in certain software**

  There is a very important thing that must not be forgotten: a web application is to be built, and that involves programming in various programming languages and web frameworks. Therefore, the developers must have a sufficient programming level in the given language in order to carry out a functional and secure development of the web application.

  There are several free sources that have been mentioned before (Udemy, Youtube, etc.) to acquire some basic knowledge of Django, Node.js, etc. (in the frameworks case) and of Python, JavaScript, CSS, HTML, etc. (in the case of web frameworks). The official pages provided by these software companies is of good help, as well as necessary.

After receiving the specific education, there is a need to carry out an **internal auditory** involving checklists, questionnaires and quizzes. The main objective is to check whether if the people involved in the application have correctly reached the required objectives.

- Checklists, which help to check if involved workers know processes may require the performance of a checklist and if they have a clear order of the procedures to take into account. Checklists are a very important task to carry out, since cybersecurity management can be complex [38].

  An example of a checklist regarding system or application management [39] could be:

1. Company security policies must be in place.
2. Security policies must be written (or re-written) and enforced along with the training process.
3. Performance of a computer software and hardware asset list.
4. Data must be classified by usage and sensitivity

- Questionnaires, another dynamic perspective to check the acquired knowledge by the person who previously took cybersecurity courses.

  Some examples can be:

  1. What are the three fundamental pillars to assure the security in information?
  2. What is the best way to keep passwords secure?
  3. How is data protected?
  4. What is Cryptography?
  5. What is a Firewall and why is it used?

- Multiple choice questions, another dynamic procedure to check if requirements are met in a more user-friendly way.

  Some examples can be:

  – Which one of the following can be considered as the class of computer threats?

    a) Information Disclosure
    b) DoS Attack
    c) Phising

d) A and B are correct

3.1.2 Security Requirements of the Web Application

Web applications are designed to perform a task, to provide a service. This is something obvious, but in the world of modern secure software development these services must be executed securely: when a new feature is added to a web application, it must undergo a process of analysis, justification and critique. If a secure software if what is wanted to be developed, it is essential to count on several (and realistic) software security requirements [40].

Requirements gathering is not an easy task, yet it must be meticulously done. First of all, the difference between functional requirements and non-functional requirements must be understood. The first of the two refers to what a system has to do and be to perform according to specifications (the service to provide following specifications), while the latter term refers to how those requirements must be met securely. It is basically a security feature to increase the user's trust in the system [41].

Carrying out a deeper analysis of the earlier terms:

1. Functional requirements include any requirement which specifies what the system should do, describing the behaviour of a particular function in the system when certain conditions are met [42]. For example, "A hammer has the ability to drive nails into wood".

2. Non-functional requirements differ from functional ones as they basically describe how a system works, how it should perform a certain action. One can think of them as quality attributes for a system, as they explain how to perform the functional requirements. Rather than focusing on

product features and on user requirements (functional), they focus on the product properties and user expectations. For example, "the computer should suspend every 15 minutes of inaction".

Both kinds of security requirements will be taken into account when gathering requirements for the web application in question. Functionalities are implemented, but they must be done securely. The methods to follow in order to list them can be very varied, particularly when it comes to listing the security requirements. This S-SDLC suggests using the STRIDE model approach. The STRIDE concept will be covered later on in the Threat Modeling section.

3.1.3 Metrics definition and Compliance Reporting for the Web Application

It is important everyone involved in the project follows the same metrics when talking about the security quality of the system. If a bug was found in the web application, it will be reported following these standards:

- The impact produced by hypothetical vulnerabilities in the system will be classified qualitatively in four different groups according to their severity: low, medium, high and critical.

- The grading will be given to the different security dimensions: Confidentiality (C), Integrity (I), Availability (A), Authenticity (Au) and Traceability (T). The mean of all this dimensions will be summed up into a final grading in a score from 0 to 10.

- The correspondence is the following:

– **Low impact** affects the system in any of the mentioned dimensions. The cyber incident could be fixed in about a working day and it can cause some private reputational damage.

  Green colour is associated to this section. The grading of this section will go from 0-3.99

– **Moderate impact** affects more than 20% of the business processes and system of the web application. The problems found in these area lead to a 5% interruption of the activity of users and it would take to fix it from 1 to 5 working days. These vulnerabilities have a higher importance than the previous ones and therefore, the reputational damage produced brings media coverage.

  Yellow colour is associated to this section. The grading of this section will go from 4 to 6.99.

– **Important impact** denotes the situation where 50% of all the process carried out are affected. The availability of the system is cut off for around 1 hour and affects more than 15% of all users. It would take to bring the situation back to normal from 5 to 30 working days. Media coverage appears and the damage produced does not only affect the reputation of the web application's organization, but also third parties.

  Orange colour is associated to this section. The grading of this section will go from 7 to 8.99.

– **Critical impact** describe the most serious situation where 75% to 90% of all functionalities are affected. In this situation, sensible

data is stolen or tampered with, involving personal data from users that would cause problems in their daily living. Service would be unavailable for 8 to 24 hours or more for a higher amount than 50% of all users. Cyber incidents of this type take between 30 to 100 working days time and, in the worst cases, even more. The reputational damages are very high and there is a continuous media coverage nationally and internationally.

Red colour is associated to this section. The grading of this section will go from 9 to 10.

3.2 Second Stage: Design

After the first stage has been performed, it comes to time to understand the possible risks the system may be exposed to. The study of all the assets belonging to the system and the way they behave is essential.

3.2.1 Threat Modeling Performance

Carrying out the threat Modeling process may be the most important step to make sure the functioning of the system is understood. The S-SDLC being designed for websites bets on elaborating a threat analysis counting on the following characteristics:

- **The threat model approach to be used will be software focused.**

  Not choosing a threat model can lead to inconsistency [43]: it is the best one can do to find problems in the design of the system.

  There are several approaches to define of the threat model including the unstructured way (brainstorming) and the structured way. In this

S-SDLC the unstructured one will not be the followed one, since it can get messy and get the members involved in the development process confused. Within the structured approach, there are three *perspectives* to choose from:

1. Focusing on assets: This may be the most intuitive approach when thinking of threat Modeling. An asset is something of value and most of the time, the model spins around things one wants to protect and things the attackers want. However, focusing on assets is not the best of the approaches. After all it is quite of *disordered* one and most importantly, there is not a direct line between assets and threats.
2. Focusing on attackers: This is a very natural approach, if it was considered there is no point on defending a system if no one would attack it. Nevertheless, it may be even worse than the assets one. In the end, risks would be humanized and it might drive people to do a lot of guesswork and speculating whether if a given risk should be taken seriously or not.
3. Focusing on software: This is the most complete and accurate approach out of all the described ones. The key to this affirmation is developers must perfectly understand the software they will be developing and there is nothing better than focusing the threat model to it. Software models may require more time than the others, but the results after it is done do benefit the whole team.

- **DFD and UML diagrams as the Type of Diagram.**

There are multiple types of diagrams within the Software-Centric Model. The most common one to be found is DFD or Data Flow Diagram. This kind of diagrams are quite useful to understand the architecture of the system and it will be applied for such purpose. It is also very interesting to use the the **UML** or Unified Modeling Language. This diagram type is more complex than DFDs, but it can prove to be very useful when presenting the threats the system me suffer. There are many types of UML diagrams and people developing a given website can use the one whichever they want. In this document, the **use-case diagram** will be the utilized one.

- **STRIDE strategy to finding threats.**

  STRIDE stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. It is an elicitation technique designed by two Microsoft engineers in the late 1990's [44], whose main objective is the given application or system fulfills the CIA triad by identifying all types of attacks that software tends to experience.

  STRIDE methodology is perfect to spot threats in the design process and it should be used along with a model (software approach), drawing DFD diagrams, etc.

  The combination of both strategies results into a great study of the web application, where developers have a strong knowledge of it and it can help them to figure out the right defenses to fight possible attacks.

- **Identifying and Mitigating risks using the defined metrics.**

  Every possible threat to the system generates a risk. This risks can be

managed following a few structured stages, which is composed of three main points:

1. Defining the risk level.

   This is a standard procedure defined in the first phase of the S-SDLC (Common Metrics). The resulting number is the product of impact times probability and will be allocated within a color according to the conditions given in each of the colors defined.

2. What actions will be taken with that risk.

   Risks can be managed in four different ways:

   (a) Avoiding the risk: This is the case where an activity is no longer carried out to have no change to suffer from a risk.

   (b) Reducing the risk: In this situation, countermeasures are taken in order to reduce the possibility of this risk coming true.

   (c) Transferring the risk: A third party is used to handle the risk for one's organization instead of doing it on its own.

   (d) Ignoring the risk: The risk is assumed and there is nothing done to mitigate it. Perhaps because the reward is less than the investment, or because there resources to fight it are not given.

3. How this actions will be taken (excluding the risk ignoring case).

3.2.2 Design Requirements Establishment

Engineers designing a secure system often make use of several security tools (cryptography, authentication controls, etc.). This tools must be used in order to meet the security requirements, but it can't get out of hand either.

We often try to make systems/applications so secure it can be difficult for users to make use of them. Therefore, design requirements come into play.

In the web applications, security requirements and functional requirements will be gathered in the first stage of the S-SDLC. In this phase, they will be used as a basis to produce the design requirements that will be the primary requirements on which the system will be based to provide the necessary services securely and safely.

3.2.3 Definition and Usage of Cryptography Standards

Cryptography is the key to protect the system and the information being treated within it. Therefore, the choice of cryptography standards requires quite a long time to be done since it is fundamental to the security involving channels through which information in sent, information storage, etc.

This section will be dedicated to explain the encryption standards that the web application will expected to have. There are several aspects to be taken into account and some of them are:

- **Web Application's Framework.**

  A web application framework is a piece of software which allows a programmer to code a web application easily, without having to code it from scratch and therefore, avoiding a lot of coding errors [45]. It includes templating capabilities for presenting information, the programming environment for scripting the flow of information and the application programming interfaces (APIs) for accessing underlying data resources [46]. Cryptography will naturally be influenced by the web framework used with which the app is developed and the security level this cryptography

provides can be improved if it was necessary.

- **Database data encryption**

  All the users' personal data must be stored within the database securely. This is where cryptography comes in to play: to avoid situations where an attacker could get access to this data in plain text. Hashing algorithms are used here, preferably having SHA-256 or higher (as SHA-1 has been proved insecure)

- **Encryption within the transportation of information**

  Implementing HTTPS is very important, as it is the best way to assure the information in the web application is encrypted while moving in the network [47]. All services must make use cryptographically secure versions of SSL/TLS [48].

  - TLS 1.2 should be enabled
  - TLS 1.1 and TLS 1.0 have to be enabled just to make sure there exists compatibility with previous versions.
  - SSL 3 and SSL 2 should be disabled.

3.2.4 Management the Security Risk of Third-Party Components

The design stage started off with the performance of a threat modeling. Threats are found and afterwards, a risk analysis is carried out. The main goal is to understand the possible problems to which a system or app is exposed to and the impact and probability this they can cause. This, however, is applied not only to the software one develops, but also to the software a developer borrows from a given source (open source or commercial) in order

to perform certain tasks regarding the development of the web application. This is known as third party software and it as important to manage its risks as it is for one's own, since a flaw in these components leads to a failure one's system.

Web applications have the following third party software components [49]:

- Web Frameworks (Django, Nodejs, PHP, etc.)
- System Components
- Web servers (Apache httpd, Nginx, etc.)
- Libraries (jQuery, etc.)

Each website will have its own and this S-SDLC bets on following a complete and in-depth guide developed by SAFECode [50] SAFECode: Managing Security Risks Inherent in the Use of Third-party Components or applying the one proposed for Open Source software by Microsoft SDL Microsoft: Open Source Security. In this document, the one used will be the Microsoft one.

3.2.5 Definition of Approved Tools

Web development tools are key actors when it turns to adapt and integrate in the development environments [51]. In this project, they represent a very important help in the continuous objective to accomplish a secure web application.

Tools can be used in several stages regarding a secure web application development process. This S-SDLC suggests using the making use of them in the following stages:

- **Digital tools for training.**

  These kind of tools include Quizziz [52], Kahoot! [53] or Socrative [54]. They will be used in the training and awareness process in the first of the phases of the S-SDLC. They are very practical and engaging, and have been proven a very effective method to learn.

  Note: The web application to be analyzed in this document is similar to these ones, but it will be ignored in this stage (the application in question is not developed at this point).

- **Threat modeling tools.**

  The threat modeling process is a complex and difficult that can be eased with the use of tools. They can either be commercial or open source. Out of the commercial ones, Microsoft's SDL Threat Modeling Tool is a very complete choice and among the open source choices, there are three important ones: TRIKE, SeaMonster and Elevation of Privilege: the game [55]. Since this S-SDLC trusts the STRIDE approach to perform threat modeling, this will be the tool used for this section.

- **IDE for developing code.**

  Frequently, programmers use text editors for code such as Sublime Text [56] or code editors such as Visual Studio Code [57]. When writing code, it is always a better idea to count with an IDE for many reasons: it has a debugging functionality, it leads to writing a higher quality code, etc. Basically, an IDE contains certain amount of intelligence that provides the developer to spend less time to write better code, allowing him or her to save resources having to look for bugs [58].

- **Tools to analyze code statically and dynamically and perform a penetration testing.**

  Code can be written in many different ways, yet not all of them may be correct to achieve a secure software. When developing web applications, it is one's interest to write code in such way the the vectors of attack are reduced.

  Doing this manually is a complex and long-lasting task. Therefore, tools have been invented in order to help developers with this issues. This S-SDLC suggests to use the following ones:

  – **To analyze third party applications, open source tools such as OWASP Dependency-Check.**

    OWASP Dependency-Check is a Software Composition Analysis (or SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a project's dependencies [59]. It checks if there is a CPE to a given dependency. Then, a report will be generated linking to the associated CVE entries. The central engine contains a series of analyzers that inspect project dependencies and collect pieces of information about the dependencies (called in-tool tests).

    Other thirds party services and data sources such as the NPM Audit API [60] are used for specific technologies.

  – **To analyze code statically, Sonarlint, SonarQube and SonarScanner tools.**

    The first is an IDE extension to detect and fix issues as code is written and the second helps providing continuous code quality,

highlighting issues found on code and looking after its health [61]. Sonarlint is the agent whose main features are bug detection, instantaneous feedback and decision-making assistance. SonarQube is the central server that processes analyses, detecting tricky issues and (most importantly) performs security analysis statically [62]. SonarScanner is a separate client type application that in connection with the SonarQube server will run project analysis and then send the results to the SonarQube server to process it [63].

– **Dynamic code analysis is another way of analyzing code and it will be discussed later. There are several helpful tools for this activity, but the main ones are OWASP ZAP (Zed Attach Proxy) [64] and Wapiti [65].**

The first of the tools has been developed by OWASP. It is an easy-to-use security web application testing tool that by means of both manual or automatic scanning helps to expose:

* Application error disclosure
* Cookie not HttpOnly flag
* Missing anti-CSRF tokens and security headers
* Private IP disclosure
* Session ID in URL rewrite
* SQL injection
* XSS injection

The other tool is Wapiti, another open source tool that finds security vulnerabilities by carrying out black box testing. It operates like a

fuzzer and provides for both GET and POST http attack methods. It helps to expose:

  * Command Execution detection
  * CRLF injection
  * Database injection
  * File disclosure
  * Shellshock or Bash bug
  * SSRF (Server Side Request Forgery)
  * Weak .htaccess configurations that can be bypassed
  * XSS injection
  * XXE injection

- **Penetration testing using Burp suite** Burp suite is a java based web penetration framework [66] and helps to identify vulnerabilities and verify attack vectors that are affecting web applications. It can be considered an interception proxy at it acts as a man in the middle intercepting traffic between the network and the user.

3.3 Third Stage: Development

The design phase is finished and it comes to time to start writing code to develop the web application. It is important coders pay as much as attention to write code as well as to make it secure. This is sometimes complicated, as code can be written in many different ways where many of the choices selected that might have looked alright turn to be insecure.

To help developers writing a more secure code when it comes to web applications, there are a variety techniques such as SAST or DAST. In this

phase of the S-SDLC the focus will be placed on the Static Application Security Testing (SAST). DAST or Dynamic Application Security Testing will be covered in future phases together with other techniques.

3.3.1 Performance of a Static Application Security Testing (SAST)

SAST analysis is introduced as a code review process where source code is scanned [67]. This kind of analysis has full access to the web application's code, covering unlinked/hidden code fragments (among others) displaying the exact line where the bug takes place. Nevertheless, they do not take into account the operating environment, web server or the database storage. Basically, there would be a solution if the web application programmer wanted to analyze code to check if there was a SSRF attack (Server Side Request Forgery attack) where the attacker can induce induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing [68]. The analyzer would scan the source code to find functions that could be vulnerable to this attack, bugs, etc. However, if what it was wanted to be checked are weak passwords or real time errors, static code analyzers are not the right choice.

This S-SDLC opts to use SonarQube for this task. The tool was explained in the previous phase.

3.4 Fourth Stage: Testing

As it is known, the earlier in the SDLC bugs are found, the better it is. In the previous phase, SAST is introduced to help developers correct web applications errors on the fly (scanning source code) and it should also be used in the testing phase. SAST analysis for web applications [69] provides with a

high scalability, detection of errors such as buffer overflows, SQL injections, SSRF as cited above, etc. and it displays very visual and helpful tips for developers. Withal, this kind of analyzer has weaknesses that are beyond its reach, such as the difficulty to find vulnerabilities (authentication problems, access control, etc.) automatically, the high number of false positives or the trouble to find flaws on compiled code.

For all these reasons, there is a need to use other options: DAST and penetration testing.

3.4.1 Performance of a Dynamic Application Security Testing (DAST)

Different from SAST, DAST is run after the code is finished and the application is running. It is also known as a behavioural analysis, where problems are found during the execution of the code [70].

This kind of analysis does not access the source code at all, but has the same resources as a normal user would have towards the web application [71]. It is what is known as black box testing. DAST analyses web application's security at runtime by means of HTTP requests, forms, links, etc. Unlike SAST, DAST does not depend on the programming language web applications use, but only needs to support support client side technologies (since it is the parts DAST can use).

If the case should arise in which an application is scanned looking for SQL Injection, the DAST tool would try to exploit the application like an automated penetration testing. If it was successful it would show the developer how it was done, but it would not display the exact piece of code where the vulnerability was found (as SAST did). DAST shows how it was done to try

to fix the problem.

This S-SDLC bets on using OWASP ZAP and Wapiti, as it was covered in the Tools Definition stage.

3.4.2 Performance of a Penetration Testing

SAST and DAST methodologies are applied to web applications to find all possible vulnerabilities it can contain. There is an additional step after DAST scanning known as penetration testing.

Web Application Penetration Testing [72] is the activity that includes a s a methodological series of steps whose main objective is to gather information about the target system, finding vulnerabilities or flaws in them, researching for exploits that will succeed against those vulnerabilities and compromise the web application. Due to the sensitive and important data web applications can contain and the continuous improvement in hackers' skills at cracking web applications, the need to add another security layer is in the SDLC is probably the most cost-effective strategy in fighting off web application vulnerabilities.

The web application penetration testings can be performed in two ways: authenticated and unauthenticated. Either way, this S-SDLC defines five steps a web application penetration testing must follow [73]:

1. Defining the scope: Before a penetration testing can take place, the scope must be defined. This way, pen testers know what areas exactly they must test in the web app.

2. Reconnaissance and intelligence gathering: This is probably the most important step in whole process, since it provides with a wealth of in-

formation to identify vulnerabilities easily and exploit them later.

3. Vulnerability discovery: In this phase, hacking tools are used to identify exploitable security vulnerabilities.

4. Exploitation: Security vulnerabilities are exploited using hacking techniques following a plan previously established in order to avoid disruption and ensuring regulatory compliance.

5. Reporting and recommendations: After the testing is completed, testers must develop a document with all findings and supply possible remediation to the problems found.

3.5 Fifth Stage: Deployment and Maintenance

The development of a we application never ends after testings are successful. Not at all. Cyber attacks are no longer things that only happen in movies. Organizations must be ready to face adverse situations where systems/applications are compromised.

Web applications are no exception to this situation. In fact, they represent 94% of all cyber attacks [74]. Web application attacks are becoming more and more complex and they are very varied. At least 80% of all web applications suffer from at least one important, critical or urgent vulnerability and 45% have more than one [75].

This data shows clearly is more than likeable that almost all web applications will suffer from a cyber attack at some point along their existence. After this assumption, there are solutions to be applied in order to make these attacks have a lower impact than they intend to have (or even none

at all). This solutions are summed up into a Standard Incident Response Process.

3.5.1 Safe Deployment Requirements

After all the necessary testing has been performed, the next step for the application is to be deployed. For this to happen, there are several factors the web application in question may require in order to be ready for its safe and operational use. Deployment requirements describe the precise, desired configuration of a software system [76], relating the non-functional (security) requirements with the functional ones.

Each web application has its own, but they are of tremendous significance: the better and the more effective these requirements are, the lesser the chances are of suffering vulnerabilities in the operational stage. It is important to remind it is way more expensive to fix new bugs or vulnerabilities in the production stage of the SDLC than in the previous ones.

3.5.2 Establishment of a Standard Incident Response Process

The probability of a web application from suffering a cyber attack are quite high as data has shown. The consequences can be quite significant and counting with an incident response plan is vital to keep the business process in action and minimize the operational risks.

The attacks can be very varied and this must be carried out in several structured steps and must be very versatile. These steps are [77] preparation, detection, contention, addressing, recovery and learning from it.

However, cyber attacks are not the only case where a system can be damaged. Other threats such as natural disasters, industrial incidents or even

unintentional errors can lead to bad situations. Thus, other plans must be carried out: disaster recovery plan and business continuity plan.

Chapter 4

S-SDLC Implementation in the E-Gamified Platform

In this chapter, the implementation of the previously defined S-SDLC will be applied to see how it would solve the problem of other students being able to access each others access. For such task, the web application will undergo every single stage of the Web Application S-SDLC.

4.1 Preparation Implementation

The first of all stages detailed in the Web Applications S-SDLC is the requirements gathering one. Three different tasks will be fulfilled throughout this phase.

4.1.1 Training and security compliance

Training and security compliance compose a very important stage in the S-SDLC, since they put the focus on improving the weakest point on any web application: humans. Having the people involved in the development of a web application go through a training process is an essential piece to assure the best possible performance of the application, both securely and effectively.

There are three training phases every single person will undergo:

1. **Web Applications' education (programming languages, fundamental concepts, etc.)**

   As is has been cited several occasions the web application is built with Django Framework, having the back-end in python and the front-end in JavaScript, HTML and CSS. Naturally, the personnel in charge of carrying out the development need to have at least a basic notion of the cited programming languages. For that purpose, I recommend checking the Freecodecamp Channel in Youtube or their official Website Freecodecamp. This is a an open source platform for people to learn programming at every level (beginner, intermediate and advanced) and it would perfectly suit anyone seeking education in the coding field.

   Django's case is different. It corresponds to the framework where the web application is developed and the education to be received about it must be up to the task. The mentioned channel works well to acquire some basic knowledge about it, yet not enough. Developers must check the official Django Website to understand deeply how it works. The web app is based in the Websocket protocol, which makes it possible to establish a persistent connection between the client and the server and both parties can start sending data at any time [78]. This provides multiple advantages, such as having bi-directional data exchange/transfer, traversing firewalls and proxies with ease, backwards compatibility with HTTP connections and publishing or subscribing event patterns [79]. Django provides a library which permits establishing this kind of connections called Channels. To get proper education in this field, checking

the Django Channels [80] website is a must.

2. **Security education**

   A survey was carried out by SoftwareSecured to over 200 developers [81]. There, several functions and features where offered to the programmers, who had to establish a priority order. Security was left overall at the bottom of the list. Security is thought to be frustrating, unnecessary for developers or even unknown for some. However, it is necessary to have some basic formation and assure security compliance in the software to be developed. Therefore, every person involved in the creation of the software must submit to all security concepts described in the web application's S-SDLC from the previous chapter.

3. **Auditory**

   Assuming every one acquired the necessary knowledge by means of the previous references, there is a need to carry out an auditory to check if the education given is effective.

   The auditory process is carried out following three different kinds of testing:

   - Checklists
     - Checklist to assure operational security in the web application.
       a) Limit access to employees applying the minimum privilege methodology.
       b) Implement SSL Data Encryption

c) Secure the Business Network (only allowing authenticated users to get in)

d) Enable 2-factor authentication on every device

e) Perform cyber security assessments regularly.

– Checklist regarding the security in data.

a) Encryption must be used wherever it may be required

b) Secure software IDE's and hardware (laptops, mobile devices, and storage devices) must be used along the development.

c) Enable automatic wiping of forgotten passwords or obligation of restoring passwords when a personal device is lost.

d) Secure Sockets Layer (SSL) in place when using the Internet to ensure secure data transfers. When there is a login in the application or when students respond to the teacher.

– Checklist regarding the need of testing [39]:

a) There is a need to establish a regular monitoring of all aspects of security regarding the application.

b) Regularly scheduled security testing within a determined period of time.

c) Scan for data types to make sure they are secure and properly stored.

d) Establish an external penetration testing to ensure your staff has not missed something after every kind of scan has been performed.

- Questionnaires

- Questionnaires regarding web security.
  a) What are the most common cyberattacks regarding web applications?
  b) Explain the STRIDE strategy.
  c) What is Information Disclosure and how can it happen?
  d) What is SSL?
  e) What hashing algorithms are considered unsafe to store passwords?
  f) What is a brute force attack?
- Questionnaires regarding django framework and WebSocket Protocol.
  a) What is a web application framework?
  b) In which programming language is Django's back-end written in?
  c) What is WebSocket protocol and how does it differ from HTTP?
  d) Can you name an example of an application that makes use of WebSockets?

• Multiple choice questions

- Multiple choice questions regarding security in general terms [82] [83].
  a) ____ is a type of software designed to help the user's computer detect viruses and avoid them.
     a. Malware

b. Adware

c. Antivirus

d. Both B and C

b) Which of the following refers to the violation of the principle if a computer is no more accessible?

a. Access control

b. Confidentiality

c. Availability

d. All of the above

c) In the CIA Triad, which one of the following is not involved?

a. Availability

b. Confidentiality

c. Authenticity

d. Integrity

d) Which of the following malware's type allows the attacker to access the administrative controls and enables his/or her to do almost anything he wants to do with the infected computers.

a. RATs

b. Worms

c. Rootkits

d. Botnets

– Multiple choice questions regarding web applications security [84].

a) If a DNS server accepts and uses the wrong details from a host that has no authority giving that information, then this technique is called ...?

   a. DNS hijacking

   b. DNS lookup

   c. DNS spoofing

   d. All of the above

   e. a and b

   f. None of the above

b) SSL stands for:

   a. Secure Software Layer

   b. Secure Socket Layer

   c. Socket Software Layer

   d. Secure System Layer

c) OWASP Number 1 element in the list is:

   a. XSS

   b. CSRF

   c. Broken Authentication and Session Management

   d. Injection

4.1.2 Security Requirements of the Web Application

The requirements gathering will be taken in separately: first will come the functional requirements (what the system must do) followed by the non-functional (security) requirements (how it should do it).

Both will be listed separately, starting with the functional requirements:

1. Both students and teacher will be able to access the web application in order to carry out their respective tasks.

2. The teacher will be able to display the questions whenever he/she desires.

3. Students will be able to answer every question sent by the teacher back to him/her.

4. Students will see their respective result when finishing with the quiz.

The security requirements will be listed following the STRIDE model. Before getting deep into the listing, the meaning of each of the letters that compose the word STRIDE and what it represents must be very clear. Hence, a table will now be displayed:

Threat	**Property**
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-Repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

Table 4.1: STRIDE Properties Table

There are several security requirements to be extracted following the STRIDE table and desired properties the system should posses to achieve a secure exchange of information:

- Authentication requirements

    1. Only authenticated users can access to the services the application provides.
    2. 2FA is not required. If so, only to access the admin panel.
    3. Only the administrator will have access to the database.
    4. Only users who posses an email with the uah.es domain will be able to create an account and access the application.
    5. Users will be automatically logged out after about 15 minutes of inactivity.

- Tampering requirements

    1. Only the administrator will be able to access the database and hence, the only person allowed to change the database fields.
    2. No one including the administrator, will have access to the users passwords.
    3. No user will be able to tamper with other's passwords.
    4. Passwords will be ciphered using safe algorithms.
    5. Data sent will travel through a secure channel that will be provide protection against tampering, as well as messages.

- Non-Repudiation requirements

    1. Logs must be stored. Django does not perform such action by default but it can be configured to do so.

2. Only the administrator will be able to access the logs information and no other user.

- Confidentiality requirements

1. The data content in the database will only be exposed to the administrator. He/She will be the only person to access it.
2. If the administrator computer is stolen, the intruder will only be able to see the administrator database if the admin had accessed it within a period of 15 minutes of inactivity. In case this computer is turned off before it is stolen, there is no chance to access the data without knowing the password (and if so, the admin account security can be strengthened with a double authentication factor).
3. Most importantly, the content of the communication between user A (the teacher) and user B (a particular student) will only be exposed to them and to no other person.
4. The information between both users will travel through a secure channel (SSL/TLS) so that it is not understandable if intercepted before it reaches the final user.

- Availability requirements

1. The application should be available 100% of the time within the lecture period. This e-gamified platform is used as a tool to help students acquire difficult concepts easier and it is indispensable to count on its availability when it is required.

Previous Django and Django channels versions where highly vulnerable to DoS attacks, but this no longer as usual in newer versions. This was categorized as a medium importance that had the network as its entry point [85].

- Authorization requirements

  1. Only administrators will be able modify user's privileges, including deleting, modifying or even adding a new user with admin privileges.
  2. The authorization is controlled in a central authorization engine.
  3. This authorization engine works accounts that belong to groups of account according to their privilege level.

  Django previous versions where vulnerable to elevation of privilege accounts, particularly one regarding an Improper Privilege Management, where a remotely authenticated user could modify the */django/contrib /admin/options.py* file to assign itself privilege accounts due to a vulnerability in the save() method. This was categorized as a medium importance vulnerability but current versions solve it [86].

4.1.3 Metrics definition and Compliance Reporting

This section is to be developed quite straightaway. The metrics definition and and the compliance reporting will be faithful to the defined procedure in the web applications S-SDLC from chapter 3.

- The study of the possible threats and the impact they can have in the application will be performed analysing every security dimension.

- Vulnerability identification, threat analysis and risk assessment will be executed qualitatively.

- The representation severity will be displayed both numerically and classified into four dimensions: low, medium, important and critical. They will be identified with colors:

Dimension	Color
Low	0-3.99
Medium	4-6.99
Importance	7-8.99
Critical	9-10

Table 4.2: Metrics Table

4.2 Design Implementation

Threat Modeling and Design Review is the next phase to carry out. Threat Modeling main objective here is to understand how the system works and the possible problems it may face. Risks assessments will be performed to the software one develops, as well as for the third party one. As what to design review concerns, this will be the phase where the most important requirements will be exposed, together with the cryptography standards and the approved tools to perform determined tasks.

4.2.1 Threat Modeling Performance

Threat modeling performance is a crucial point that will include four different stages that will be carried out following the web application S-SDLC.

4.2.1.1 Threat Model Approach

The threat modeling approach will be focusing on software. The main objective here is to aim developers to understand the application the best they can. Having a good understanding of the software foundations is very helpful to understand the threats the system may be exposed to and how hard they can affect the application.

4.2.1.2 Application's Architecture and Use Cases

DFD and UML will be used in order to carry out such tasks. The first one is used to define the architecture of the application and to understand how it works. The second one will be used to display a possible use case scenario where an attack takes place. However, this makes more sense after carrying out the STRIDE strategy to finding threats and will be used later on.

The architecture of the web application is the following when the student sends information to the teacher (and vice versa) the process goes as follows:

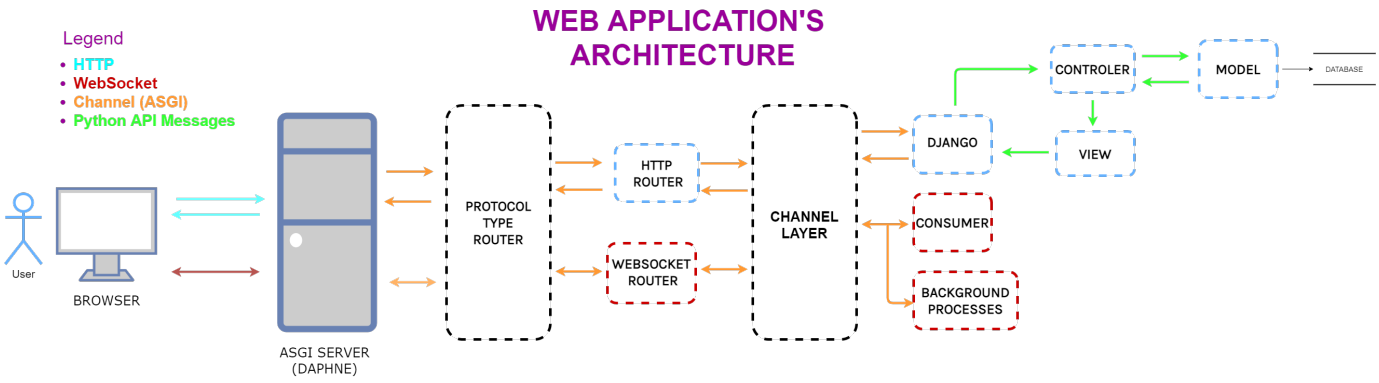

Figure 4.1: Web Application System

It is important to understand what the elements in the image are, the processes taking place and the information flow.

- The user in question makes a request to the back-end, in this case an **ASGI server**. This request can be an HTTP one (the blue arrow), which is unidirectional, or a web socket one (the red arrow), which is bidirectional.

- The **ASGI server** (Asynchronous Server Gateway Interface) is a common standard by which applications servers and applications talk to each other asynchronously [87]. It accepts both asynchronous and synchronous code and usual used server is Daphne.

- The data leaves the ASGI server through a **channel**, which is like a pipe. It allows Django to handle web sockets same way it did with http requests [88]. This channel message reaches the **Protocol Type Router**, which a general router that decides to which router handler the message will be send according to the type of incoming message (WebSocket or http).

- Next, the router the messages reach the **channel layer**, which allows different instances of an application to talk with each other [89]. It is useful to avoid accessing the database anytime a message has to be send from one point to another without having to access the database.

- The channel layer hands in the message to the consumer or to a normal django handler. A **consumer** is defined as the smallest channel unit, a tiny application that consumes an event taking place. Background processes take place as well.

- In the case of the Django common http requests, its architecture is the

common **MVC** (Model, View and Controller) with each of them having its own responsibility:

- The **Controller** receives the the request and communicates with the model seeking for data in the database.
- The **Model** retrieves the data from the database and hands it back to the **Controller**.
- The **Controller** gives the information to the **View** which will deploy the desired information to the user.

The communication between these parties is done through Python API messages.

4.2.1.3 STRIDE strategy to finding threats

Web applications as a general concept can suffer from all the STRIDE threats. For that reason, it is important to perform a DFD as it was done previously to be able to understand to which particular attacks it may be vulnerable to. In order to help the task of finding all possible threats, the Elevation of Privilege Game will be used in this section.

Hereunder all of the threats that would affect the exchange of information between students and teacher will be listed following the STRIDE order:

- **Spoofing threats**
  - The attacker (let us say Eve) could spoof the teacher so that all answers are sent to her. To perform this, the attacker could carry out a phishing attack where she connected the teacher to a false front end and steal her/his credentials. From that point onwards,

Eve could perform a MITM attack and see all answers written by the students.

- The attacker could spoof a student to then eavesdrop the answers others respond. This could be performed the same way as the teachers case.
- The attacker could try to brute force an administrator's, teacher's or student's account to take over it without being stopped by the web application.
- The administrator account comes with a default admin:admin credentials and does not force a change. If Eve stole this account, she could later tamper with the questions the teacher sends.

- **Tampering threats**
  - Tampering threats reduce themselves in this case to network tampering ones. Related to the previous covered threats, the attacker Eve could redirect the information flowing through the network to a machine under its control after performing a MITM attack.
  - An attacker could be able to manipulate data flowing in the network (from student to teacher and vice-versa) since there is not data protection on it's flow.

- **Repudiation threats**
  - The attacker Eve could take over someone's account and perform some fraudulent action (an example in this case could be responding a student's wrong answers on purpose). The student in question

would say he/she did not do such thing, but there would no way to prove it.

  - Following the previous line, if Eve was accused of doing whatever she has done she could say she did not do it and there would be no way to prove it wrong. This could be because the system has no logs regarding that or because the logs do not capture the required information.

- **Information Disclosure threats**

  - Threats of this type will once again focus on data flow over the network by performing MITM attacks. The attacker could be able to read the data on the network because there is no cryptography at all used to cipher the traffic. Data could be read in clear text.
  - The attacker could read all the information in a channel because it is not encrypted (http).
  - If the traffic was in fact ciphered, the attacker could learn from analyzing traffic.

- **Denial Of Service threats** Note: DoS threats can affect both clients and server and the attacker Eve can be unauthenticated or authenticated (if she has previously stolen credentials).

  - The attacker could craft network requests to overwhelm the front-end and not being able to perform any kind of action.
  - The attacker could send enough network requests to exhaust the server resources to handle network traffic. The server would be left

unavailable.

- The users would not be able to use the application, since the service would be stopped due to the exhaustion of the network resources.

- **Elevation of Privilege threats threats**
  - The attacker Eve could try to perform some injection of code to perform actions she is not allow to.
  - The attacker authenticated as a user could try to give herself admin privileges to then perform malicious actions that could affect network data flow.
  - The attacker could perform some kind of XSS to later perform some malicious action.

4.2.1.4 Risk management

The risk management will be executed in a table, where the three points described in the *Identifying and Mitigating risks using the defined metrics* part in page 45 will be included.

The risks to be covered will follow the line of Chapter 4 putting its main focus on data flow.

- Spoofing Risks

Threat	Risk Importance	Action	Mitigation (if so)
Forge/Steal Keys	6.5	Transfer the Risk	The risk is transferred to Django Framework. Support will use OS tools to secure storage and generate keys securely.
Spoof Endnode	5	Mitigate the Risk	Cryptography will be used to assure correct authentication.
MITM Attacks	8.1	Mitigate the Risk	Achieve strong authentication using cryptography
On-Path Injection of Packets	8.5	Mitigate the Risk	Use of message or cryptographic channel authentication to avoid Eve seeing and inserting her own packets
Blind Injection of Packets	7	Mitigate the Risk	Use of message or cryptographic channel authentication to avoid Eve learning things and injecting her own packets

Table 4.3: Spoofing Threats Risks Table

- Tampering Risks

Threat	Risk Importance	Action	Mitigation (if so)
Tampering with message integrity	8	Mitigate the Risk	The adverse effects depend absolutely on the data flow. Add integrity controls in the code or tunneling.
Tampering with weak channel/ message integrity	7.8	Mitigate the Risk	Avoid using weak cryptographic algorithms such as MD5
Tampering with time and ordering of messages integrity	6.2	Ignore the risk	Replay attacks, Reflection attacks and Collisions can take place. There no other choice than trusting Django Framework support.
Tampering with upstream insertion	9	Mitigate the Risk	Insert input validation where messages could be inserted.

Table 4.4: Tampering Threats Risks Table

- Repudiation Risks

Threat	Risk Importance	Action	Mitigation (if so)
Admin account takeover	3.5	Mitigate the Risk	Assure strong authentication and logging
Normal user account takeover	2	Ignore the Risk	None
Message transaction denial	4	Mitigate the Risk	Use strong hashing algorithms logging
Message altering	4	Mitigate the Risk	Use logs combined with a strong hashing algorithms

Table 4.5: Repudiation Threats Risks Table

- Information Disclosure Risks

Threat	Risk Importance	Action	Mitigation (if so)
Information disclosure by observing a message or channel	9	Mitigate the Risk	Use cryptography to avoid outsiders view one's messages
Information disclosure with MITM attack	8.1	Mitigate the Risk	See MITM Spoofing table (4.3)
Information disclosure by observing side channels	3.2	Ignore the Risk	None
Information disclosure cause by effects of other actions	0.5	Ignore the Risk	None

Table 4.6: Information Disclosure Threats Risks Table

- Denial of Service Risks

Threat	Risk Importance	Action	Mitigation (if so)
DoS against a process	8	Ignore the Risk	None
DoS to incapacitate endpoints	5	Ignore the Risk	None
DoS against devices	8.1	Ignore the Risk	None
DoS with crafted messages	7	Mitigate the Risk	Use strong cryptography and integrity checks provided by the framework

Table 4.7: Denial of Service Threats Risks Table

- Elevation of Privilege Risks

Threat	Risk Importance	Action	Mitigation (if so)
Cross-domain issues	2	Ignore the Risk	None
Issues with design	4.5	Mitigate the Risk	Performance of tests during the design of the app
Input validation	3.5	Mitigate the Risk	Being careful with the design and check possible flaws of injections

Table 4.8: Elevation of Privilege Threats Risks Table

4.2.2 Design Requirements Establishment

This section's objective is none other than describing the main requirements among the ones said so that the security controls do not go too hard on users (making it hard for them to use the application). Therefore, the most important requirement will be set clear in this short section: the main objective is to avoid a non authorized person (a student or an external attacker) having access to the other students answers when responding or even to the teachers questions sent to them either if he/she does not belong to the class.

4.2.3 Definition and Usage of Cryptography Standards

The factors to take into account where very well described in entry 3.2.3 of Chapter 3. The development of this section will follow the same order.

4.2.3.1 Django as a Web application framework

Django is a web application Framework written in Python. It counts with certain plugins that provide with cryptography capacities to the framework and it can be obtained by installing the package *django-cryptography* [90]. This includes the possibility to apply cryptography in the back-end, for signing and key generation or for the cryptography salt.

4.2.3.2 Database data encryption

The S-SDLC recommended the use of strong encryption algorithms, such as SHA-256. This web applications counts on it, thanks to the Django. The encryption algorithm it uses to store passwords securely in the database is PBKDF2 (Password-Based Key Derivation Function 2) combined with a SHA256 algorithm [91]. PBKDF2 is a key derivation function with a sliding

computational cost whose main objective is to reduce the range of brute force attacks [92].

4.2.3.3 Encryption within the transportation of information

Django, as any web a framework allows to activate SSL/TLS to perform secure exchange of information between users. To direct all HTTP requests to HTTPS, set SECURE_SSL_REDIRECT to True in settings.py file within the django application [47].

4.2.4 Management the Security Risk of Third-Party Components

The Open Source software gives developers a lot of benefits [93], which has made them highly dependent on them. That brings (of course) a whole lot of security risks to an organization's system or application and they are known as thirds party risks.

To face them, four steps must be taken alongside the S-SDLC:

1. **Carry out an inventory of the open souce tools used**.
2. **Perform a security analysis**, including checking out for public vulnerabilities and performing static security analysis using commercial or other open source tools (destined for that purpose).
3. **Keep open source software up to date**.
4. **Carry out a security response process**.

The first two steps will be carried out using OWASP Dependency-Check , which is a Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a project's dependencies

[59]. It checks if there is a CPE to a given dependency. Then, a report will be generated linking to the associated CVE entries.

The tool is available in the previously provided link and it can directly be run in the command prompt. For this project, Ubuntu has been used as the base system and the tool must be downloaded according to the determined operating system.

A report called *dependency-check-report.html* is generated after the analysis is complete. The report first displays the tool version, followed by the time when the report was generated and the number of scanned dependencies. Then, the number of vulnerable dependencies, the found and suppressed ones are seen.

The third step is trivial an the fourth is carried out in the latest step of the S-SDLC, as described in page 58. They are both related to the maintenance of the application.

This process does not directly imply the project could be affected by other vulnerabilities. Django (the framework used in this case) makes use of plenty of Python libraries that may suffer from vulnerabilities caused by bugs they can contain. Any piece of software is never 100% secure and even more if there is a dependency in the supply chain.

4.2.5 Definition of usage of approved tools

The tools used to for this web application will be the ones described in section 3.2.4. Some of them have already been used and others will be utilized in the following sections when arriving to static and dynamic analysis.

4.3 Developing

The static analysis of the code is carried out using a set of three tools: Sonarlint, Sonarqube and SonarScanner. Each of them plays an important role when it comes to analysing some code statically and all of them will be described. Images of the actual code of the web application using the mentioned tools will be shown accompanying the explanations to make it easier to understand.

SonarLint is installed as a plugin in Pycharm IDE. The installation process is very simple ans just takes a few seconds and once it is done, it starts working right away. SonarLint analyses the code and displays the bugs marked with numbers and giving a short explanation in the first place.

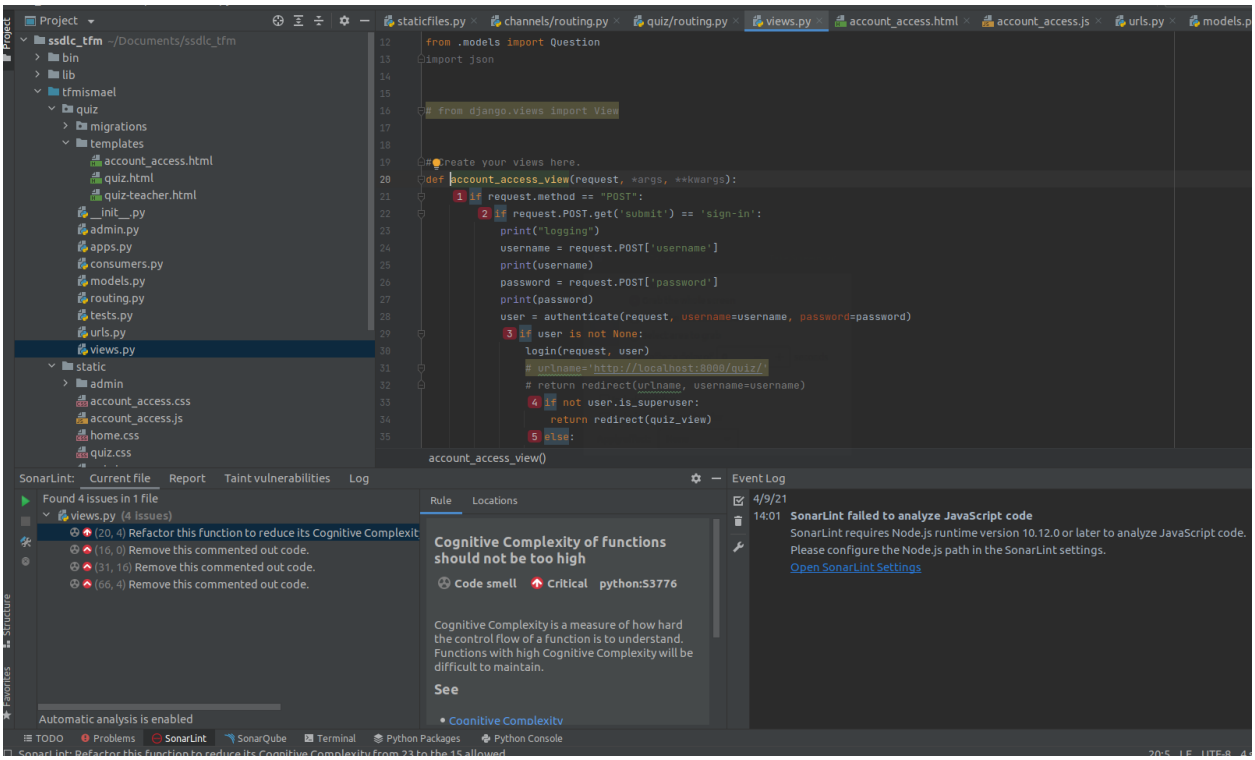

Figure 4.2: SonarLint Plugin functioning

If the user wanted to dig deeper into the explanation of why the bug has

been detected as so, an extensive description can be seen by clicking in the in the bug correction as it shown in the following picture:

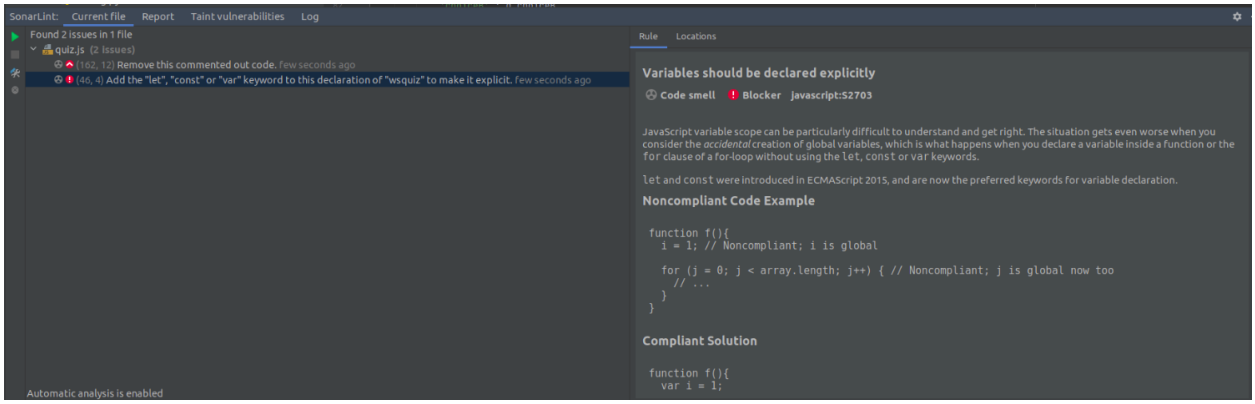

Figure 4.3: SonarLint Extensive explanation

If one pays attention to the previous image, bug description corresponds to a JavaScript file. This is something not available only with Pycharm's SonarLint Plugin, as it is thought for python language. For this case, there is a need to run SonarQube ans SonarScanner.

SonarQube can be downloaded from its official website. It must be installed and configured and it must count with OpenJDK (Open Java Development Kit, a free and open-source implementation of the Java Platform, Standard Edition SE) [94]. Next, the binary file is launched and it can be accessed from the browser introducing the url *http://localhost:9000* and introducing the default credentials (admin:admin). At this point, SonarScanner is launched. It can be downloaded from SonarQube's official website as and it must be installed and configured based on the project properties.

After it is downloaded, it is launched together with SonarQube.

4.4 Testing: Security Testing and Code Review

The security testing stage comes right after the code has been developed and statically analyzed. The section testing here is carried out in two stages: dynamic testing and penetration testing.

4.4.1 Dynamic Testing

The dynamic testing is executed using the free open source tool OWASP-ZAP, explored in previous sections. The analysis takes place during the process of usage of the web application (dynamic). The tool was previously covered in chapter 3. If wanted to check further specifications, one can be check in this website.

For this project, a Kali machine will be used to deploy the analysis. ZAP is installed by defaulta and it can be launched from the console by typing *owasp-zap*. The procedure is very simple, as in this phase an automated analysis started. It looks like the following:

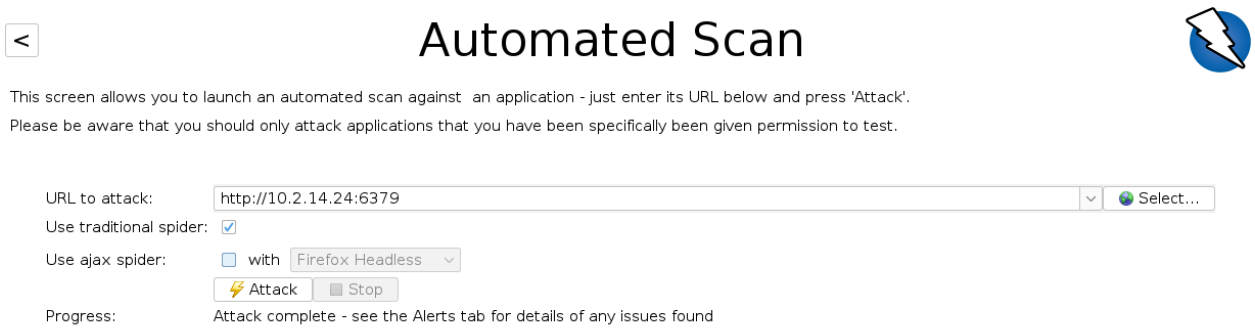

Figure 4.4: OWASP-ZAP Analysis

The results will be further discussed together with the others in the following chapter.

4.4.2 Penetration Testing

Following the line of the dynamic analysis, the penetration testing takes place while the application is launched and several tools are used to perform it. In this project, several tools are used and the results will be further described. This stage is also very important not only because it helps the organization to take several measures in the code to make it secure (as well as with static and dynamic analysis), but because it simulates a real attack that would take place simulating an attacker-like approach and it is the last barrier of the S-SDLC before its operational use.

4.5 Deployment and Maintenance

After the project has been built and passed all the determined tests the application would be ready to be deployed for its use. Nevertheless, it is obvious (even more after coming to this point) that the web application is exposed to threats every day and the organization in question needs to be prepared to face such problems. For this reason, there is a need to carry out security response plans to define the path one should follow to achieve an effective recovery after an adverse situation.

4.5.1 Safe Deployment Requirements

The testing has been performed and the application is ready to be deployed for its safe and operational use. The ones for this application are the following:

1. **A requirements.txt file.**

   This is a very important file where the libraries and packages used in

the project are used:

```
python3>=3.7
virtualenv==20.7.2
django==3.2.7
channels==3.0.4
redis==5.0.7
django-otp==1.0.6
qrcode==7.3
```

2. **The use of SSL and TLS to establish HTTPS.**

   This was told in the in the security requirements, but is must be remarked. In the testing sections, both in the static and the dynamic ones the vulnerability of HTTP POST requests (where input fields exist) in extremely clear how dangerous it is not to count with either of these protocols. Nevertheless, the respective versions need to be considered secure to avoid attacks, such as CRIME to TLS 1.0 and 1.1 (with segmentation).

3. **The DEBUG option of Django project must be set to False.**

4. **Code obfuscation**

   A web application is composed of a front end and a back end. In this case, django has been used. The back end code (the one regarding the server) is written in python and is not accessible, whilst the front end code (regarding the client) is written in JavaScript and it can be seen in

the the browser. Therefore, it is important to perform some obfuscation to the JavaScript code so that attackers do not obtain any clue from whatever code they read.

5. **Use of a Web Application Firewall (WAF) to protect the web application**

   A web application firewall (WAF) protects the web application by inspecting and filtering traffic between each web application and the internet [95]. It protect web applications from attacks such as cross-site request forgery (CSRF), cross-site-scripting (XSS), file inclusion, and SQL injection.

4.5.2 Establishment of a Standard Incident Response Process

At the end of Chapter 3, the need of counting on an incident response plan was stated, as well as the likelihood of a web application of suffering from a cyber attack. be a cyber attack in any form to the web application. This would lead to adverse situations including stealing of information, tampering with its content or even an attack that would make the service provided by the application unavailable. Accordingly, two recovery plans will have to be executed:

1. **An Incident Response plan or IRP.**

   This plan is based on a set of instructions to help ICT personnel detect, respond to and recover from network security incidents, such as this situation (suffering a cyber attack) [96].

2. **A Business Continuity Plan or BCP.**

This is a logistical plan that an organization must follow to recover and restore functions that are critical to the organization. This plan is carried out after suffering an undesired interruption and this case is a clear example of it (critical information is destroyed/lost to function) [97].

4.5.2.1 Incident Response Plan

Before starting to carry out the plan, there are requirements that need to be fulfilled:

- The objectives to be reached with this plan need to be set very clear (business processes, regulations, etc.).
- The processes involved in the plan need to have an actual and an objective maturity level assigned.
- The security director needs to have a complete understanding of the flaws the application has in order to understand the capabilities and how they can improve.
- The organization needs to count with sufficient personal and technological resources in order to execute the plan in the most effective way.

If these requirements are satisfied, the plan is ready launched. This plan is carried out following certain steps:

1. The organization must count with a *guide* that includes multiple case scenarios such as information destruction, information theft, etc. and the steps to follow in order to proceed to the fastest and most efficient recovery of such information.

2. In case an incident takes place within the web application, one should have clear who else is affected by such situation: third parties, clients, etc.

3. If one wants a plan to be effective, it must undergo a revision every now and then. This is the same that happens with the web application's code, where static and dynamical tests are executed to assure maximal security.

4. The web application counts on five security levels regarding what and who is/are affected by the possible cyber attack. If an attack was hazardous enough to provoke a data breach on sensitive information or a long lasting denial of service, it would affect individuals, groups and departments very likely (levels 1, 2 and 3), but it could also affect the organization internally and externally (reputation) (levels 4 and 5), having a possible important effect on the business process.

5. Scalability is very important in serious cases such as a cyber attack. The problem needs to be communicated fast and effectively enough to every worker affected by the adverse situation (in case it is detected).

6. Metrics and standards need to be defined and known by everyone. Every single worker needs to speak *the same tongue* and this is vital to recover faster from a problem.

7. There must be a group of experts in cybersecurity to check whether if the plan is correctly developed, the possible alerts popping up in a SIEM and how to deal with them, risks, etc.

8. Every single person needs to have a basic knowledge in cybersecurity and understand how to deal with possible cybersecurity breaches, so they can respond effectively in an adverse scenario.

4.5.2.2 Business Continuity Plan

Its main objective is to reduce the risks after a disaster and to improve the chances of recovering from an adverse situation minimizing the impact and damages suffered.

Before carrying out the business continuity plan, there is one thing that must be set clear: a BCN must fulfill certain characteristics to be considered effective. These are the following:

- A BCN must be flexible to be able to adapt to any risk emerging in the organization.
- It must be able to respond to every situation, describing the concrete actions to be followed, as well as resources used and people in charge of leading the given actions.
- Communication needs to be fluent within developers and between organization and client to avoid misunderstandings that could lead to problems when it comes to the business process.
- It must be clear who takes decisions and how they are taken when an incident takes place.

Having this in mind, the business continuity plan for this web application must count with the following stages:

1. Responsibilities must be clearly defined within team members developing the application.

2. One must define a Business Continuity Policy to establish the general framework that is necessary to ensure the implementation of an efficient and effective Business Continuity Management System.

3. It is necessary to conduct a business impact analysis, also known as BIA, is necessary to understand how the business processes carried out in the company can be affected by an incident such as a cyber attack.

4. It would be necessary to carry out a criticality analysis to know the probability and impact that an incident (cyber attack in this case) would have in the web application's organization. An essential step in a business continuity process is the definition of recovery objectives, setting the key indicators that help to start a recovery plan. One must also be clear about the RTO (Recovery Time Objective) and RPO (Recovery Point Objective) of the business recovery plan.

5. Logically, one must develop a continuity plan (as the name of the plan itself indicates) and a strategy in which there is an architecture, processes and procedure based on the objectives that have been designed for the recovery plan.

6. One must pay attention to the cost of the recovery plan. This cannot be too high as it can be counterproductive.

7. The same as with the incident response plan, one must have qualified

personnel to be able to carry out a plan of this caliber, of critical importance to recover the web application from an adverse situation.

Chapter 5

Results

This chapter is destined to study the measurements taken after the S-SDLC has been applied to the web application and understand how it has improve its security after its use and how the different phases of the Software Development Lifecycle (SDLC) have improved thanks to the application of the Secure SLDC.

5.1 Preparation results

This stage is fully dedicated to the preparation of the people involved in the web application development to learn how to use or the way the specific software that will be used for the development. The main objective is to get the people ready enough to face the design process and have a remarkable critical opinion to take decisions based on based on substantiated ideas.

The idea of the S-SDLC here is to first **train** the worker not only in the software development world, but also in the security one. Out of all the resources to cover, reading the book threat modeling helps the person to have a better understanding of many key points when it comes to developing a software securely. The concepts acquired in this training are then leveraged with an internal auditory, which is the definitive step to check if the knowledge gained is sufficient.

The **requirements gathering** comes right after. The effort of carefully

reading and analysing the Threat Modeling book starts paying off. The Web application's requirements are gathered using the STRIDE's model proposed in the book. This is a very methodical approach, since it makes it more visual for the reader to see what requirements need to be met in each of the security dimensions. About twenty requirements were gathered in total in this particular project and it gives developers very important hints about what the system should do and how it should do it.

A clear definition of the **standard metrics** to be used along the development process is vital so that every person involved in the web application development process speaks the same tongue. To make it the most intuitive possible, four colours have been used: green, yellow, orange and red. They are separated by ranges where risks are allocated based on the impact and probability and the conditions given according to each color.

5.2 Design results

This is the section dedicated to give the requirements an applicability and start a high level design of the web application.

The first of the stages is the **threat modeling** one, which is key to understand every single element of the system. It studies threats and vulnerabilities that can put the system at risk. The chosen approach was software focus and STRIDE to finding threats. The architecture of the system was drawn using DFD diagrams and then risks were analysed. After the risk analysis, one has a clear idea on what things need to be implemented in the development stage (the next one) to make the system more secure. Out of all the possible upgrades, two have been carried out: password hardening (more strict policies)

and two factor authentication for the administrator's account.

The **design requirements** were then established, followed by the **cryptography standards**. Django provides PBKDF2_SHA256 encryption algorithm, which is quite secure. If the administrator wanted to check some user's password, this is what will be seen:

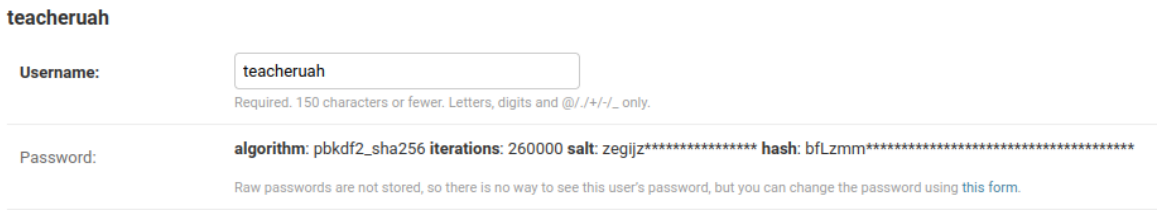

Figure 5.1: User's hashed data

This is considered secure and the passwords are safely stored protected by a strong protection algorithm.

The next stage is the **management of the third party risks**. It is important to check whether if the web application is affected by a certain vulnerability coming from a third party. For this purpose, there are open source tools such as Dependency check which help solving this issue. The tool is available in the link provided in page 84 and it can directly be run in the command prompt. For this project, Ubuntu has been used as the base system and the tool's performance looks like the following:

Figure 5.2: OWASP Dependency Check Running

Once it is done, it generates an html report called *dependency-check-report.html* which has the coming image:

Figure 5.3: OWASP Dependency Check Report

As seen in the image, the total number of vulnerabilities found in this web application is 0, which means the project does not contain any third party vulnerability (allegedly). Other details can be studied if one wanted by clicking in the Summary link. There all the possible vulnerable dependencies are displayed.

The last stage in this section is the definition of approved tools. This phase requires some research to find the tools that have been proven to be secure and at the same time, that best adapt to the system necessities.

5.3 Development results

This section is dedicated to develop the code to improve the system functionalities according to the previous steps results. Two new functionalities have been included to the system, while a static analysis is carried out to ensure the security in the code.

5.3.1 Password enforcement policies

The password enforcement is something necessary to be carried out. Following the threat analysis conclusions, the password strength is something vital to avoid an attacker stealing some student's credentials, as well as the teacher's (which is even worse because they have direct access to everyone answers and results). Thus, there has been an initiative to force users to create their accounts using strong passwords which must fulfill:

- The password must be at least eight characters long.
- The password must contain at least one lowercase letter.
- The password must contain at least one uppercase letter.

- The password must contain at least one special character

- The password must contain at least one number.

To have a visual representation of what it looks like when someone wants to create an account, some pictures are now attached.

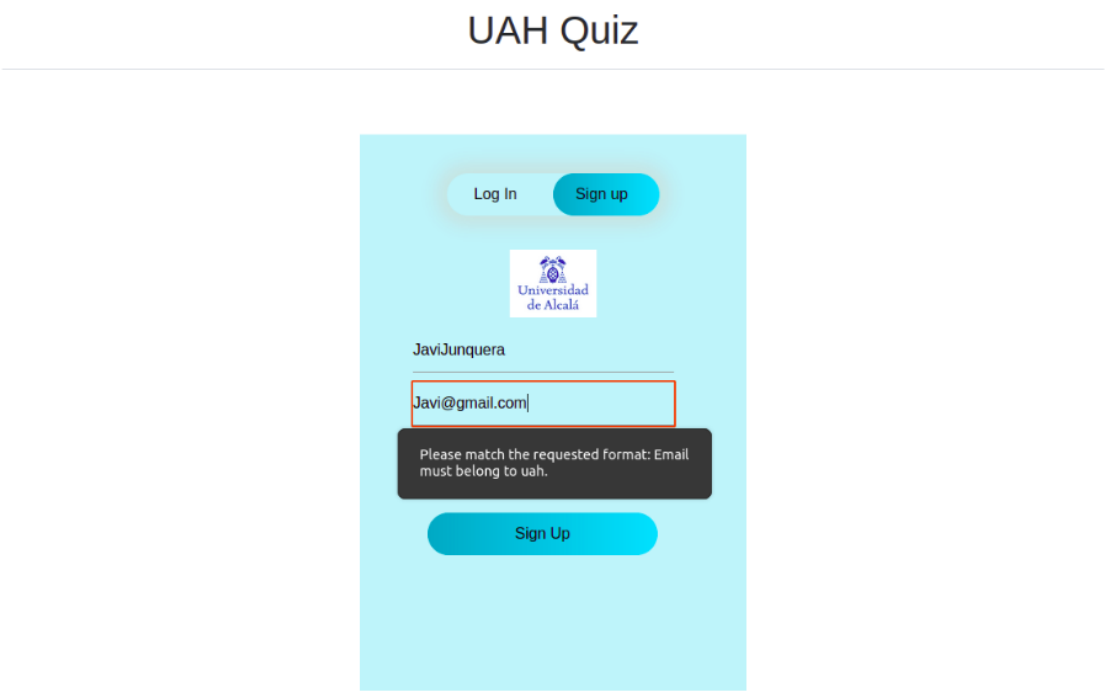

Figure 5.4: Email checking

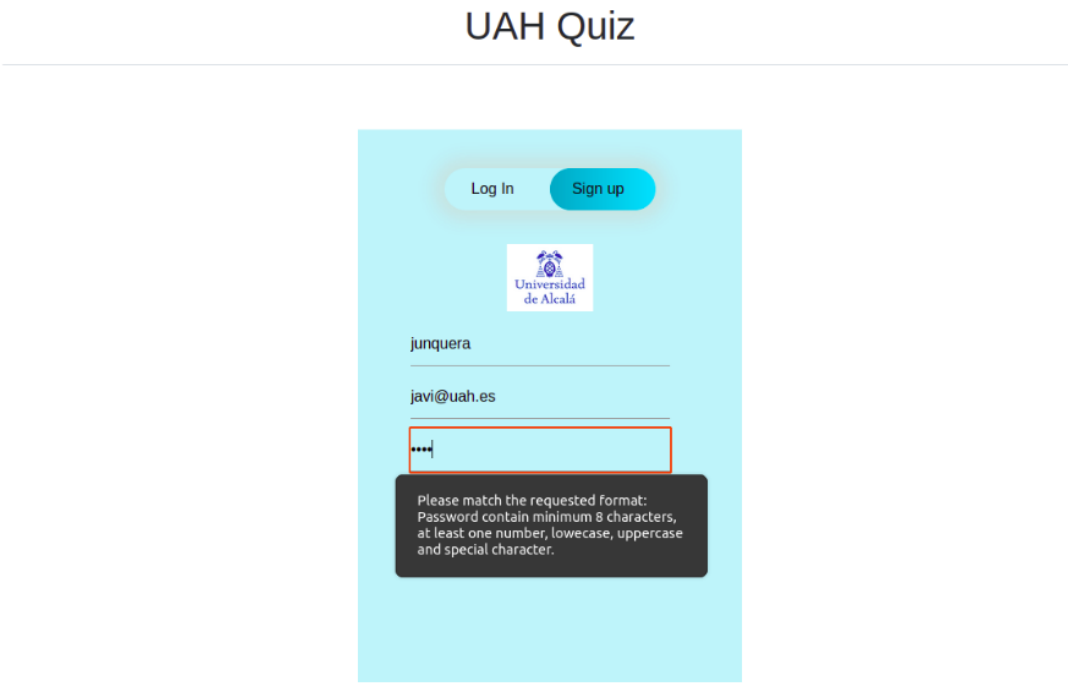

Figure 5.5: Password checking

As it can be seen, there is no other way to access but by having an email belonging to the university and having a strong password. After they log in, all users can participate in the quiz and their results are safely seen in the teacher's account.

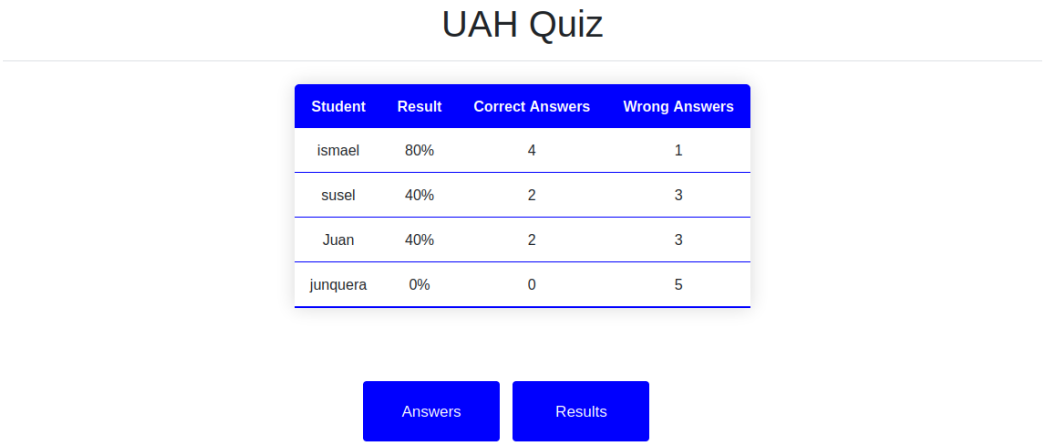

Figure 5.6: Results in the web application

5.3.2 Implementation of a two factor authentication system (admin)

The administrator's account must be extremely protected. Whoever steals his/her credentials, has access to all questions and their correct answers in the database. Also, the attacker could see the the responses each student. Therefore, a two factor authentication system has been developed to give an extra layer of security to these accounts.

In order for this to work, two python libraries need to be installed (django-otp and qrcode). Also, the django application code needs to be adapted and the Google Authenticator Tool is used. This tool implements two-step verification services using the Time-based One-time Password Algorithm (TOTP) and HMAC-based One-time Password algorithm (HOTP), for authenticating users of software applications [98]. It can be configured to whatever one desires.

The whole process has been captured with images so one has a clearer view of how it works and enforces the security of the system.

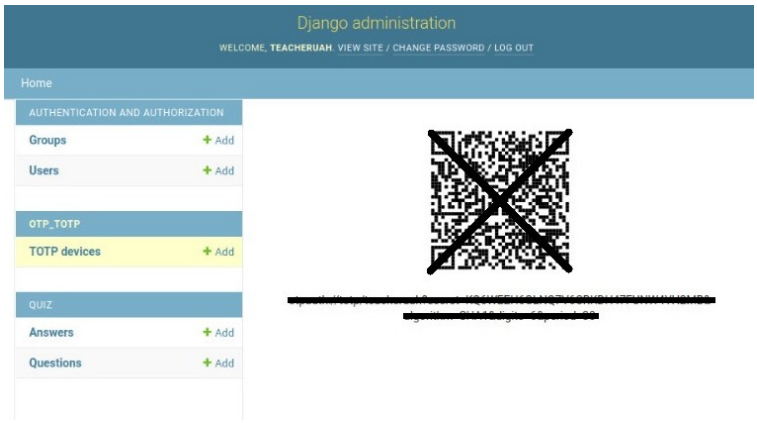

Figure 5.7: QRcode generation

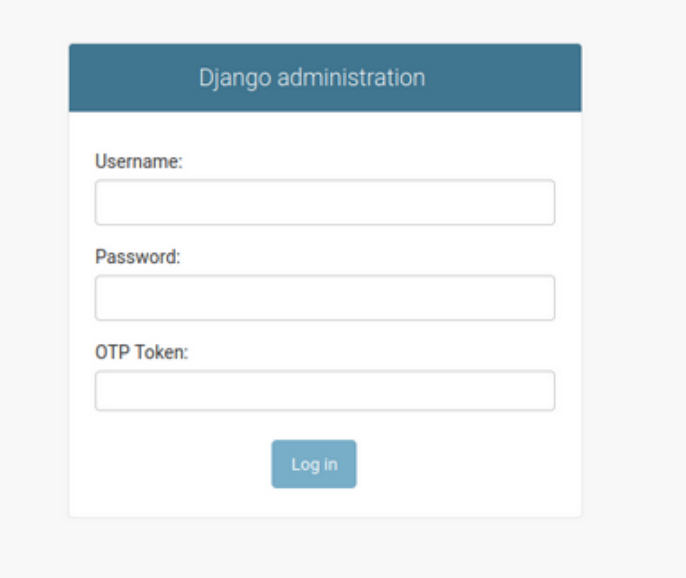

Figure 5.8: Token Required to log in

The required token can only be seen in the administrator's phone application Google Authenticator, which is regenerated every 25 seconds.

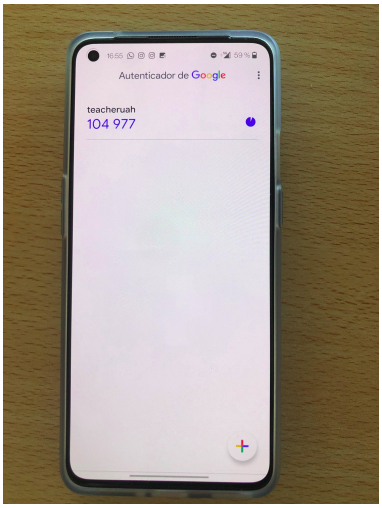

Figure 5.9: Token Required generated in the phone

5.3.3 Static analysis

The static analysis is carried out using three different tools: Sonalint, SonarQube and SonarScanner. In the previous chapter, one could see how

SonarLint displays errors on real time (with more or less detail) of Python code. However, if one wanted to check static code of JS or CSS SonarQube and SonarScanner need to be launched. These two provide a more exhaustive analysis of the possible errors found in the whole project.

An analysis is carried out and the results are displayed in the url **http://localhost:9000**, corresponding to the SonarQube server. Once it is launched, the user must introduce its credentials.

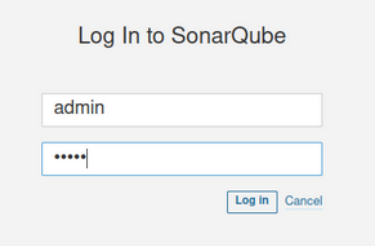

Figure 5.10: SonarQube Login

SonarScanner must also be started. For this to work, the configuration file of SonarScanner must be configured introducing several variables, such as the user and password of the SonarQube server, the project key and name and the path of the project that is wanted to be analysed.

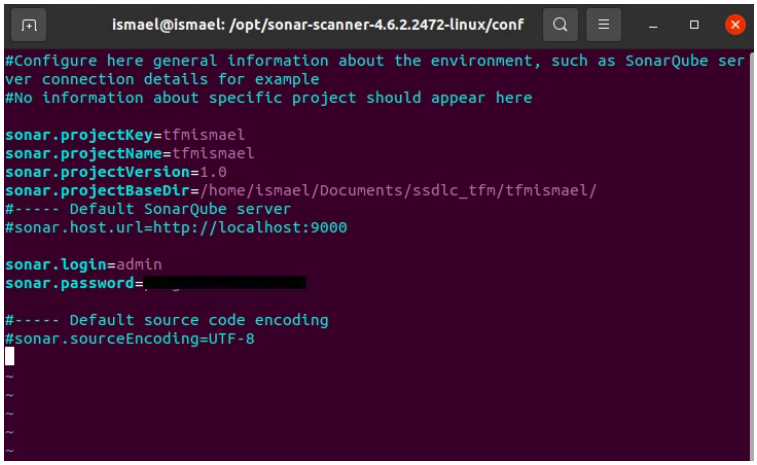

Figure 5.11: SonarScanner Configuration

After SonarScanner is launched and the project is analysed, the results are displayed in the SonarQube url. Several bugs, vulnerabilities, security hotspots and code code smells are found. They are shown as follows:

Figure 5.12: Several Bugs

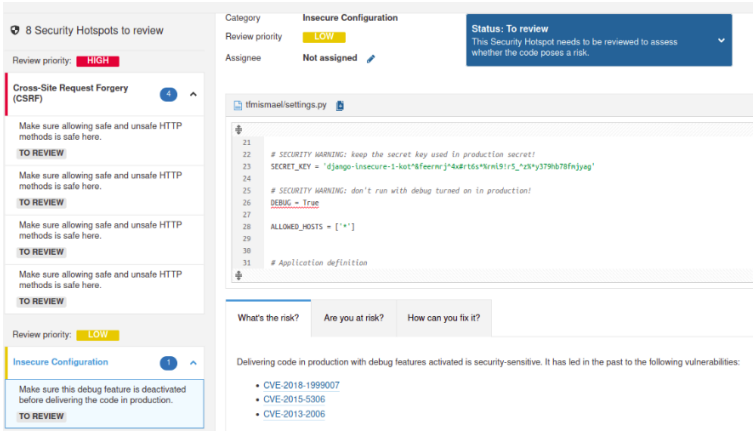

Figure 5.13: Security Hotspots

The corrections can be carried out following the instructions given by SonarQube, which are very clear and intuitive.

The overall results are displayed in the project section. The structure is as follows:

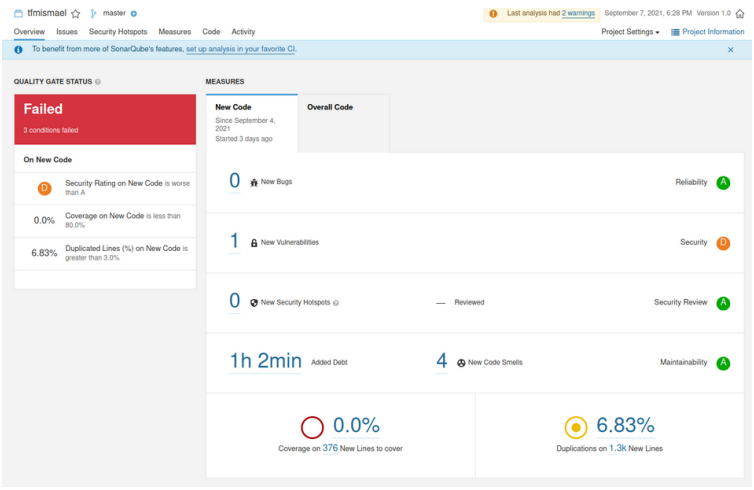

Figure 5.14: Results in SonarQube

First, a red box with the word *Failed* can be seen. This means the web application is considered insecure after performing the static analysis with SonarQube. On the right, the number of bugs, possible vulnerabilities and security hotspots (sensitive pieces of code that must manually be reviewed) and other data. In this case, the bugs and vulnerabilities all corresponded to python files django imported, not from the actual web application files coded by developers. This image was taken after a few corrections, but they can be improved and the number of errors will decrease:

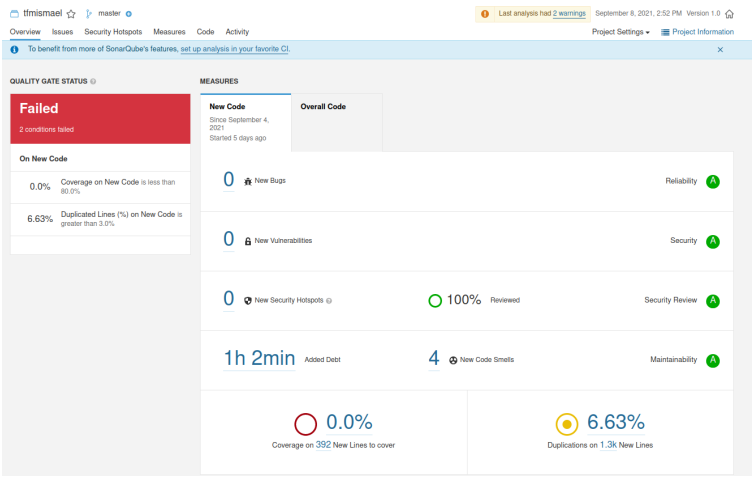

Figure 5.15: New Analysis

One of the corrections was against a possible CSRF attack. This was solved by applying a decorator before a python function (request_safe).

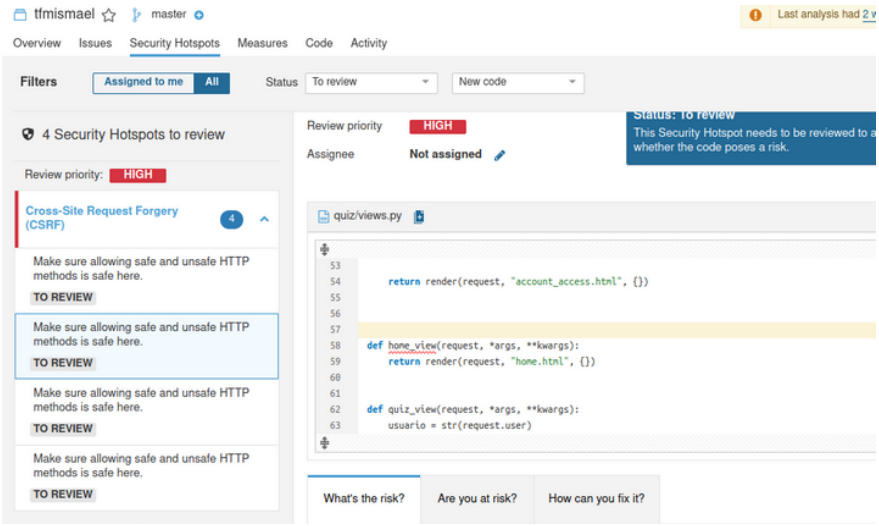

Figure 5.16: CSRF possible attack

```
@require_safe
def home_view(request, *args, **kwargs):
    return render(request, "home.html", {})
```

Figure 5.17: CSRF possible attack solution

Even though the project is shown as insecure, this is due to two conditions that fail in the analysis which are the percentage % of new covered code (which is non relevant because the one regarding self written code can be checked using SonarLint and the other belongs to files imported by django) and the percentage of duplicated lines (again belonging to django imported files. This can be supported with a static analysis performed only to the application (not to the whole django project).

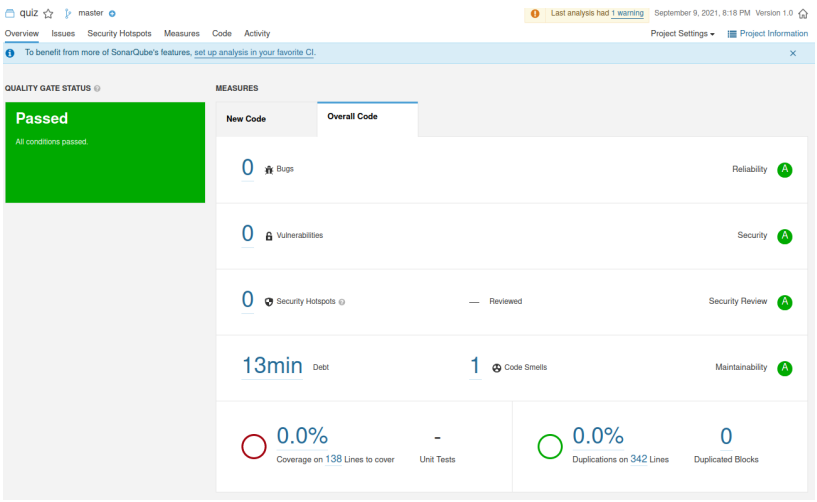

Figure 5.18: Analysis to the quiz application

As it can be seen, now SonarQube says the quiz application is secure and passes all examined factors with an **A** mark. Only one code smell (it may be hard for maintainers to correct possible errors in that section of the code) is shown (but still gives maintainability the highest mark).

After the results are obtained and analysed, it comes to time to proceed to the next stage.

5.4 Testing results

As it is known, a web application is never ready to be put out in the world without first carrying out a testing phase where a dynamic analysis and a penetration testing are carried out.

5.4.1 Results of the dynamic analysis

The dynamic analysis was performed, as said in the previous chapter, using OWASP zap automated scan. Once it is run, a few errors have been detected. Some pictures of a few of them will be displayed.

- **Cross-Domain JavaScript Source File Inclusion**

This error alerts the user that there are some scripts loaded from a third party site and its integrity is not being checked.

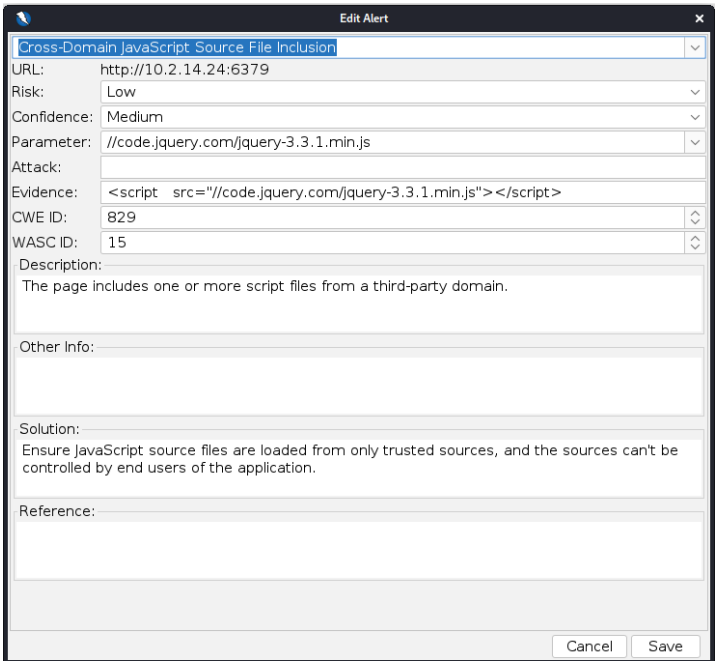

Figure 5.19: Cross-Domain JavaScript Source File Inclusion

In order to solve it, one must check the integrity of the urls from which this files are loaded. There are several websites, such as SRI Hast Generator, that perform a hash of the url (SHA-256 or SHA-384) and must be introduced in the source file as follows:

Figure 5.20: Cross-Domain JavaScript Source File Inclusion

- **X-Content-Type-Options Header Missing**

This error tells the user the option Anti-MIME-sniffing was not set to true, allowing old version of some browsers to examine the content of a particular asset [99]. This is can be performed to JS and CSS files in old versions of web browsers and most of all, in Internet Explorer.

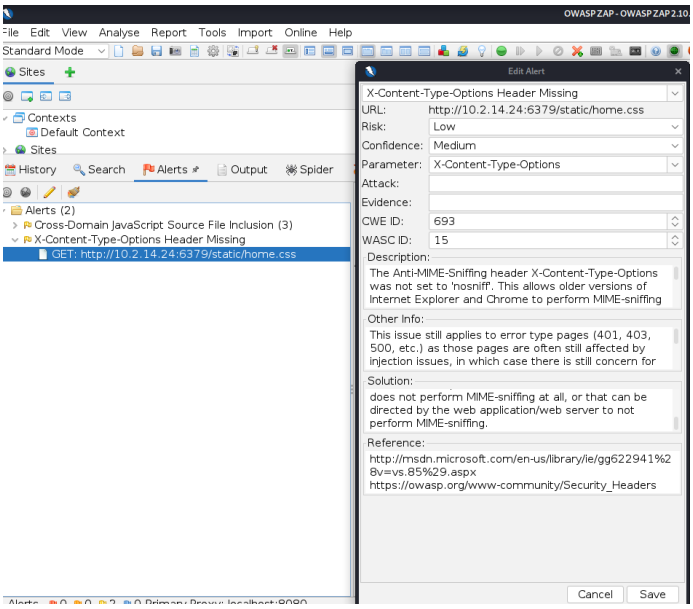

Figure 5.21: X-Content-Type-Options Header Missing

To fix it, one should use updated versions of web browsers and avoid using Internet Explorer. In the django settings file, the following variable should be defined and set to true:

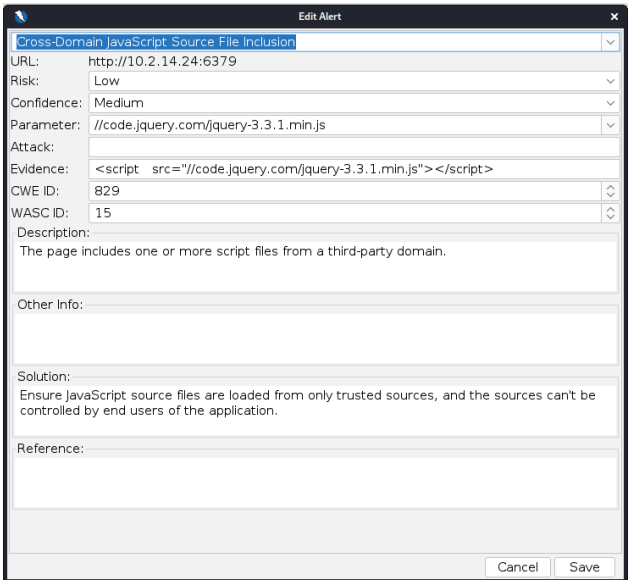

Figure 5.22: Solution to X-Content-Type-Options Header Missing

- **Information Disclosure from comments**

  Some of the comments written in JS files may give an attacker hints in case the code is intercepted that can provide critical information to understand how to code works.

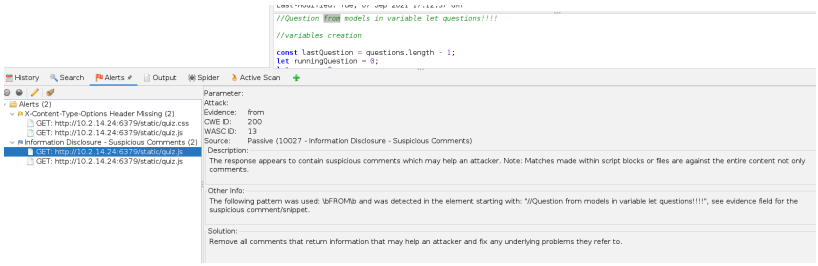

Figure 5.23: Suspicious comments that must be avoided

This comments should be deleted and the code must be obfuscated, as stated in the Deployment stage.

OWASP has also been used for spidering to find hidden files in the application, but none of importance has been detected.

5.4.2 Penetration testing evidences

The web penetration testing stage was executed after the web application as another complement to the other two analysis to test the possible vulnerabilities the web application could be exposed to after the whole S-SDLC process. In this case, Burp Suite has been used to perform a Brute force attacks, and other tools for other checkings.

5.4.3 Burp Suite Brute Force attacks

- Performing a brute force attack using Burp Suite starts with turning on the proxy interception to catch the requests sent from the browser. Since http is used (which is very insecure in post requests where data is introduced) the text intercepted is in clear text.

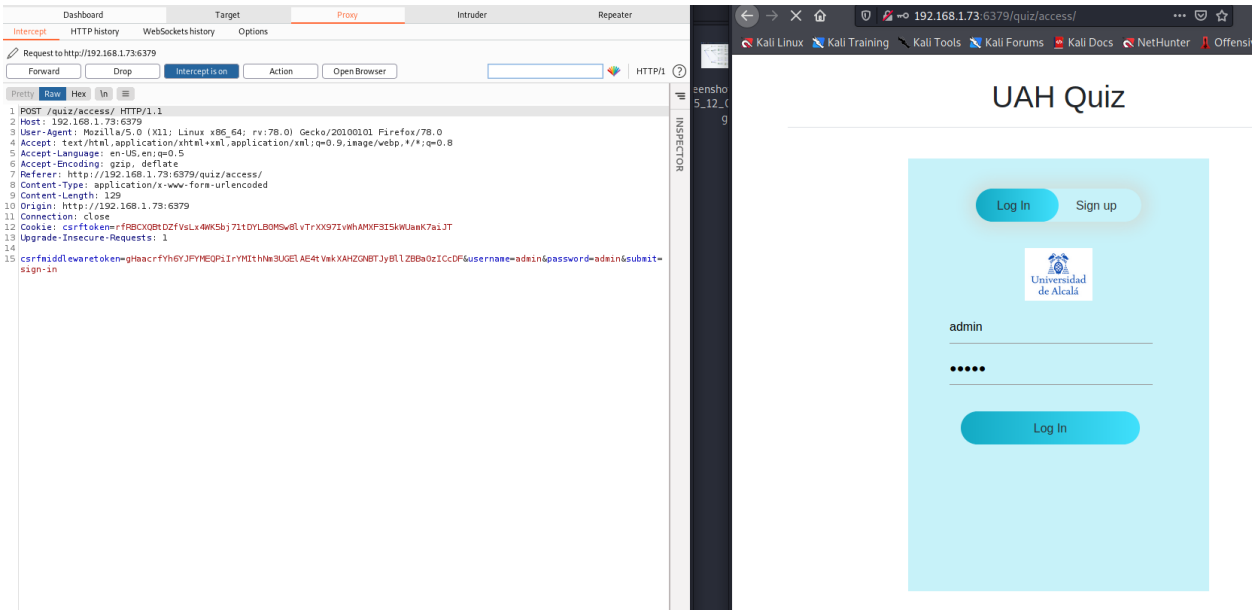

Figure 5.24: Caught introducing text

- The variables that want to be intercepted are sent to the intruder (which is another module inside Burp Suite). In this case, they are admin and

admin (user and password). A cluster bomb attack (brute force of more than one variable) is launched. Here, several combinations will be tried.

- This will be a simple attack where a list of names and passwords will be introduced. A list of very common user names and people common names in Spanish will be introduced, as well as the most used passwords.

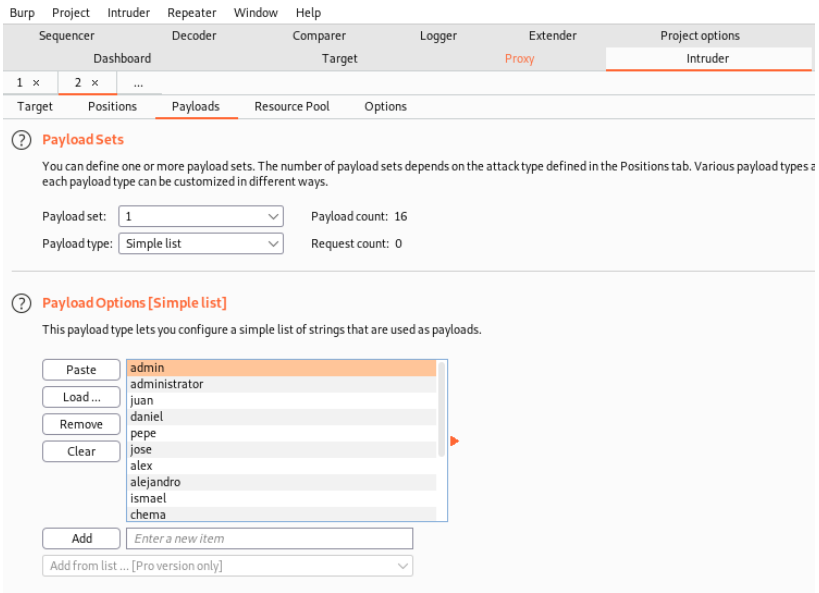

Figure 5.25: List of users

- After both lists are introduced and the attack is launched, Burp Suite starts combining all possibilities, until the correct one is found.

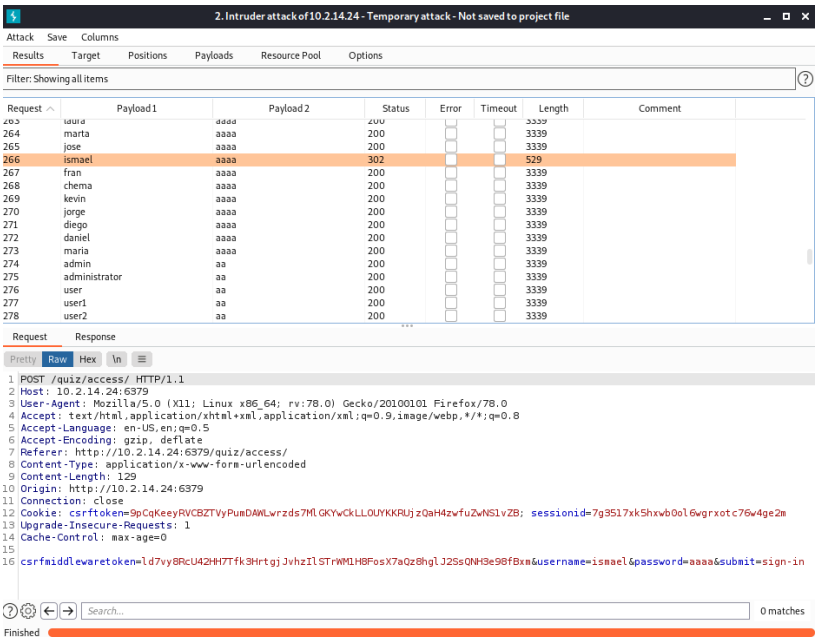

Figure 5.26: Found name and password

Note that the status of the response is 302, which stands for found. This is a clear indication that tells developers the importance of using strong and secure passwords, as well as encouraging the establishment of a double factor authentication.

5.4.4 Other tools usage

The use of a WAF increases the protection of a web application and its use is extremely recommended. The ones that do not use it are more likeable to suffer from an attacks and that is of much interest for attackers. Therefore, tools such as **wafw00f** pre-installed in Kali Linux are used to check if the given web application counts with any of them.

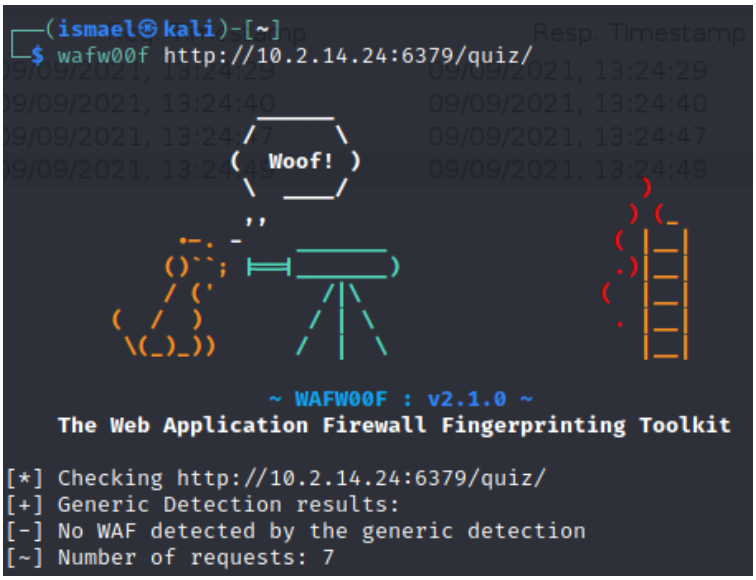

Figure 5.27: WAF detection

As it can be seen in the image, no WAFs are detected and that leaves the application vulnerable.

5.5 Deployment and maintenance results

Once the testing phase is ended, the application is ready to be put out in the world for its use, but this does not mean the secure development lifecycle is ended. Not at all.

First, several safe deployment requirements need to be gathered. These will be necessary to fulfill if one wants to launch a web application safely. Five requirements were listed in this project, starting with the requirements.txt file (which is imperative for others to know the required libraries to work in the development of the application), the establishment of SSL and TLS by a certified entity to http (so that data flows securely and it needs not much explanation), setting the debug variable to false (only set to true locally so that developers have an easier time finding errors), the obfuscation of code

(so that the front end code cannot be plagiarized) and the use of a WAF to keep the web application more protected.

In case an incident took place leading to an adverse situation, a standard incident response plan needs to be developed. The objective is to have every body understand the way the whole organization needs to response when this happens. Both the Incident Response plan or IRP and Business Continuity Plan or BCP contain the sufficient steps to minimize the provoked damage.

5.6 Closing Lines

Every phase on every stage of the S-SDLC is very important and must be done meticulously in order to correct problems as earliest as possible. The objective: saving time and money.

Taking care of a appropriate preparation before getting hands on the project as so is highly important. Every person's knowledge regarding security and other software used to build the application need to be sufficient, as well has having clear what the system has to do and the way it should do it to do it secure. The metrics established need to be easy to understand and this project has tried to fulfill this task by using colors to make the process intuitive.

Threat analysis is one of the most (if not the most) important stages in the secure software development lifecycle. Several methodologies can be used to approach the gathering of possible threats and the STRIDE one proposed by Adam Shostack. These threats can come from one's software or from third party one and the risks produced by any need to be properly handled. The tools and cryptography standards need to be previously analysed to ensure

security in the application when making use of them.

While the code is being developed, it needs to be analysed with statically (static application security testing or SAST) to explore the possible weaknesses and flaws on real time. To do this task, SonarQube has been thought as the best choice to achieve the most secure code on real time. It has been proved as an exceptional option. But what after the code is finished? Is the system ready for production? Of course not: it needs to undergo a testing phase. Code testing is done applying a dynamic analysis (dynamic application security testing or DAST) to the application to assure a good security while it is running, like a real world example. An extra layer of testing is added with a penetration testing simulating an attacker point of view and making the testing more complete.

The project ends with a few clear and structured requirements needed to be achieved before the application is ready to be put out in the world for its use. Without them, the application is not warranted to be protected. In fact, the cycle never ends: it needs to be maintained and for that, a plan is required. The standard incident response plan needs to be known by everybody, so its legibility and coherence must stand out and these two factors have been the key point followed when developing it.

The results have shown how important it is to count with the convenient security measures and the earlier they are implemented, the better. For this to happen, every stage of the S-SDLC must be meticulously done achieving the best possible balance between service and security to provide the best possible software.

Chapter 6

Conclusions

The software development process is a complex issue which has traditionally been used to provide a piece of software focused only on the service it provides. Software has become a fundamental piece of our lives, evolving at an amazingly fast speed together with technology. The growing number of attacks produced by these factors has led to a shift in the way the world thinks about the SDLC, considering security as an important factor. The main aim of this project it to ensure the reader considers not only security important, but as vital as providing a service is. If a service is not secure, then there is no point on supply it.

Time ago, security was slightly introduced in the SDLC as another phase with not much importance. Today, this has been proved as an absolutely wrong approach. Security in a software must be provided all along its development process, introducing it in each stage of the SDLC as what is known as the S-SDLC (Secure Software Development Lifecycle). The process of developing an S-SDLC is not a very flexible one when it comes to adding stages.

This project is a clear example of how security can be directly included into every single phase of the SDLC, having each of the security stages contributing different things. All of them converging into one single purpose,

making a service work secure. The S-SDLC is has come and it is here to stay.

Nevertheless, researches can be carried out to make certain stages better. After all, there is always future work that could be carried out to improve investigations. The possible future work regarding this project could be:

- Modifications inside a given stage according to one's point of view.
  - Making use of different tools where to write code (different to Pycharm).
  - Making use of a different approach to STRIDE in the finding threats process. This may lead to find disparate results (other additional) to the ones found.
  - Additional static analysis and dynamic analysis software would be used in the development and testing sections. The same applies to the penetration testing activity.
- Another interesting activity would be assigning the web application a domain regarding the University of Alcalá. Giving the web application the possibility to count on SSL protection by a certification authority would solve many problems and would be a huge step to make the application operational.
- Other functionalities could be added to the application, making it a more complete application both students and teachers could use for a wide variety of activities. These functionalities would be added following every single stage of the S-SDLC defined in this project. The key to make keep the web application secure.

Bibliography

[1] S. F. Melián. (2018) Aplicación de nuevas tecnologías para el apoyo a la docencia en un entorno interactivo.

[2] S. Fernández Melián. (2018) Nueva aplicación multiplataforma para el aula invertida. prueba piloto despliegue.

[3] I. Jiménez Castro. (2020) Web sockets implementation in the application for the flipped classroom.

[4] Microsoft, "Microsoft Security Development Lifecycle," 2021. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl

[5] M. Yost, "A Brief History of Software Development," Nov. 2019. [Online]. Available: https://medium.com/@micahyost/a-brief-history-of-software-development-f67a6e6ddae0

[6] Oxford. Oxford languages and google - english | oxford languages. [Online]. Available: https://languages.oup.com/google-dictionary-en/

[7] M. Bacon. What is security? [Online]. Available: https://searchsecurity.techtarget.com/definition/security

[8] T. D. Edmiston. What is software security? [Online]. Available: https://medium.com/the-framework-by-tangram-flex/what-is-software-security-e03a5ee7a6b5

[9] Techopedia. What is software security? [Online]. Available: http://www.techopedia.com/definition/24866/software-security

[10] G. McGraw. What is software security? it's not security software. [Online]. Available: https://www.synopsys.com/blogs/software-security/software-security/

[11] R. Gibb, "What is a Web Application? | How a Web Application Works," May 2016. [Online]. Available: https://blog.stackpath.com/web-application/

[12] I. E. Team, "What Is a Web Application? How It Works, Benefits and Examples." [Online]. Available: https://www.indeed.com/career-advice/career-development/what-is-web-application

[13] Rapid7. Software development life cycle (SDLC): A security perspective. [Online]. Available: https://www.rapid7.com/fundamentals/software-development-life-cycle-sdlc/

[14] Wikipedia, "Software development," Apr. 2021, page Version ID: 1019201951. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Software_development&oldid=1019201951

[15] A. Altvater, "What Is SDLC? Understand the Software Development Life Cycle," Apr. 2020. [Online]. Available: https://stackify.com/what-is-sdlc/

[16] R. Half, "6 basic SDLC methodologies: Which one is best? | Robert Half," Jan. 2021. [Online]. Available: https://www.roberthalf.com.au/blog/employers/6-basic-sdlc-methodologies-which-one-best

[17] E. Mougoue. (2017) Ssdlc 101: What is the secure software development life cycle? [Online]. Available: https://dzone.com/articles/ssdlc-101-what-is-the-secure-software-development

[18] Microsoft, "Microsoft Security DevOps," 2021. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/devsecops

[19] S. Poremba. (2018) Security in web apps. [Online]. Available: https://securityintelligence.com/your-web-applications-are-more-vulnerable-than-you-think/

[20] P. Technologies. (2020) Positive technologies: 82 percent of web application vulnerabilities are in the source code. [Online]. Available: https://www.ptsecurity.com/ww-en/about/news/82-percent-of-web-application-vulnerabilities-are-in-the-source-code/

[21] Owasp. OWASP foundation | open source foundation for application security. [Online]. Available: https://owasp.org/

[22] OWASP.org. OWASP top 10:2021 (DRAFT FOR PEER REVIEW). [Online]. Available: https://owasp.org/Top10/

[23] D. Sveikauskas. Security by design principles according to OWASP. [Online]. Available: https://patchstack.com/security-design-principles-owasp/

[24] E. Borges. SecurityTrails | security through obscurity. [Online]. Available: https://securitytrails.com/blog/security-through-obscurity

[25] S. E. Team, "What is the secure software development life

cycle (SDLC)? | Synopsys," Jul. 2020. [Online]. Available: https://www.synopsys.com/blogs/software-security/secure-sdlc/

[26] Microsoft, "Microsoft Security Development Lifecycle Practices," 2021. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl/practices

[27] J. Fruhlinger, "Threat modeling explained: A process for anticipating cyber attacks," Apr. 2020. [Online]. Available: https://www.csoonline.com/article/3537370/threat-modeling-explained-a-process-for-anticipating-cyber-attacks.html

[28] Synopsys, "What Is SAST and How Does Static Code Analysis Work? | Synopsys," 2021. [Online]. Available: https://www.synopsys.com/glossary/what-is-sast.html

[29] A. Phadke, "SAST vs. DAST: Whatâ€™s the difference? | Synopsys," Mar. 2016. [Online]. Available: https://www.synopsys.com/blogs/software-security/sast-vs-dast-difference/

[30] Veracode, "Dynamic Analysis Security Testing (DAST)," 2021. [Online]. Available: https://www.veracode.com/security/dast-test

[31] P. Technologies. (2021) SSDL implementation. [Online]. Available: https://www.ptsecurity.com/ww-en/services/sdl/

[32] Ptsecurity. (2021) How to approach secure software development. [Online]. Available: https://www.ptsecurity.com/ww-en/analytics/knowledge-base/how-to-approach-secure-software-development/

[33] YouTube. [Online]. Available: https://www.youtube.com/

[34] Online courses - learn anything, on your schedule | udemy. [Online]. Available: https://www.udemy.com/

[35] D. S. Dan Bergh Johnsson, Daniel Deogun. (2019) Secure by design. [Online]. Available: https://www.manning.com/books/secure-by-design

[36] M. Cobb. Add threat modelling to your web application security best practices. [Online]. Available: https://www.computerweekly.com/tip/Add-threat-modelling-to-your-Web-application-security-best-practices

[37] A. Shostack. Threat modeling: Designing for security. [Online]. Available: https://threatmodelingbook.com

[38] G. Mutune. 22 best items for a cybersecurity checklist. [Online]. Available: https://cyberexperts.com/cybersecurity-checklist/

[39] S. Nasiri. Cybersecurity audit checklist. [Online]. Available: https://reciprocity.com/cybersecurity-audit-checklist/

[40] J. Boote. What are software security requirements? | synopsys. [Online]. Available: https://www.synopsys.com/blogs/software-security/software-security-requirements/

[41] I. Global. What is security requirement? [Online]. Available: https://www.igi-global.com/dictionary/discovering-core-security-requirements-drm/26121

[42] U. Eriksson. Functional vs non-functional requirements - understand the difference. Section: Requirements. [Online]. Available: https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

[43] A. Shostack. (2014) Threat modelling, chapter 2. [Online]. Available: https://moodle.ufsc.br/pluginfile.php/2377555/mod_resource/content/2/Threat%20Modeling.pdf

[44] F. Donovan. (2021) What is STRIDE and how does it anticipate cyberattacks? Section: Cloud Security. [Online]. Available: https://securityintelligence.com/articles/what-is-stride-threat-modeling-anticipate-cyberattacks/

[45] Intelegain. What are web frameworks and why you need them? [Online]. Available: https://intelegain-technologies.medium.com/what-are-web-frameworks-and-why-you-need-them-c4e8806bd0fb

[46] M. Luna. What is web development framework (WDF)? - definition from WhatIs.com. [Online]. Available: https://searchcontentmanagement.techtarget.com/definition/web-development-framework-WDF

[47] J. Jose. Encryption & hashing in django. [Online]. Available: https://www.linkedin.com/pulse/encryption-hashing-django-jerin-jose

[48] BruceCowper. Microsoft SDL cryptographic recommendations - security documentation. [Online]. Available: https://docs.microsoft.com/en-us/security/sdl/cryptographic-recommendations

[49] Vaadata. Are you using vulnerable third party code in your website? [Online]. Available: https://www.vaadata.com/blog/are-you-using-vulnerable-third-party-code-in-you-website/

[50] SAFECode. (2021) Safecode: Computer & network security. [Online]. Available: https://safecode.org

[51] M. G. Byrd. Top 10 tools to facilitate web development. [Online]. Available: https://www.searchenginewatch.com/2020/09/25/top-10-tools-to-facilitate-web-development/

[52] (2021) Quizizz — the world's most engaging learning platform. [Online]. Available: https://quizizz.com/

[53] (2021) Kahoot! | learning games | make learning awesome! [Online]. Available: https://kahoot.com/

[54] (2021) Socrative. [Online]. Available: https://www.socrative.com/

[55] A. Shostack. (2020) Elevation of privilege (EoP) threat modeling card game. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=20303

[56] (2021) Sublime text - the sophisticated text editor for code, markup and prose. [Online]. Available: https://www.sublimetext.com/

[57] (2021) Visual studio code - code editing. redefined. [Online]. Available: https://code.visualstudio.com/

[58] J. P. Mueller. (2021) Why IDEs are important for python program-

ming. [Online]. Available: https://www.dummies.com/programming/python/ides-important-python-programming/

[59] OWASP. OWASP dependency-check project. [Online]. Available: https://owasp.org/www-project-dependency-check/

[60] Npm. npm-audit | npm docs. [Online]. Available: https://docs.npmjs.com/cli/v7/commands/npm-audit

[61] Stackshare. (2021) SonarLint vs SonarQube | what are the differences? [Online]. Available: https://stackshare.io/stackups/sonarlint-vs-sonarqube

[62] Stackoverflow. (2021) SonarQube and SonarLint difference. [Online]. Available: https://stackoverflow.com/questions/39828609/sonarqube-and-sonarlint-difference

[63] S. Szołkowski. How to use SonarScanner to assure code quality. Section: Development. [Online]. Available: https://blog.setapp.pl/how-to-use-sonarscanner

[64] OWASP ZAP zed attack proxy | OWASP. [Online]. Available: https://owasp.org/www-project-zap/

[65] Wapiti : a free and open-source web-application vulnerability scanner in python for windows, linux, BSD, OSX. [Online]. Available: https://wapiti.sourceforge.io/

[66] pentestgeek.com. What is burp suite - pentest tool description.

Section: Tools. [Online]. Available: https://www.pentestgeek.com/what-is-burpsuite

[67] T. A. Nidecki. Static code analysis in web security. [Online]. Available: https://www.acunetix.com/blog/web-security-zone/dynamic-static-code-analysis-web-security/

[68] PortSwinger. What is SSRF (server-side request forgery)? tutorial & examples | web security academy. [Online]. Available: https://portswigger.net/web-security/ssrf

[69] OWASP. Source code analysis tools | OWASP. [Online]. Available: https://owasp.org/www-community/Source_Code_Analysis_Tools

[70] Technopedia. What is dynamic application security testing (DAST)? [Online]. Available: http://www.techopedia.com/definition/30958/dynamic-application-security-testing-dast

[71] T. A. Nidecki. Dynamic program analysis in web security. [Online]. Available: https://www.acunetix.com/blog/web-security-zone/dynamic-static-code-analysis-web-security/

[72] S. Stankovic. Web application penetration testing: Steps, methods, & tools. [Online]. Available: https://purplesec.us/web-application-penetration-testing/

[73] Redscan. Web application penetration testing service. [Online]. Available: https://www.redscan.com/services/penetration-testing/web-application-testing/

[74] Imperva. (2014) Web attacks: The biggest threat to your network. [Online]. Available: https://www.imperva.com/docs/ds_web_security_threats.pdf

[75] D. R. Staff. 80% of web applications contain at least one security bug. [Online]. Available: https://www.darkreading.com/vulnerabilities---threats/80--of-web-applications-contain-at-least-one-security-bug/d/d-id/1328233

[76] M. Kupfer and I. Hadar. (2012) Understanding and representing deployment requirements for achieving non-functional system properties.

[77] Z. Banach. Incident response steps in web application security. [Online]. Available: https://www.netsparker.com/blog/web-security/incident-response-steps-web-application-security/

[78] M. Ubl and E. K. C. . Y. b. m. n. s. t. f. i. t. article. Introducing WebSockets: Bringing sockets to the web - HTML5 rocks. [Online]. Available: https://www.html5rocks.com/en/tutorials/websockets/basics/

[79] Pusher. What are WebSockets? [Online]. Available: https://pusher.com/websockets

[80] Django channels — channels 3.0.4 documentation. [Online]. Available: https://channels.readthedocs.io/en/stable/

[81] S. Koussa. 6 reasons why developers don't write more secure

code | blog. [Online]. Available: https://www.softwaresecured.com/why-dont-developers-write-more-secure-code/

[82] Javapoint. Cyber security MCQ (multiple choice questions). [Online]. Available: https://www.javatpoint.com/cyber-security-mcq

[83] examradar.com. Cyber security MCQs - computer science. [Online]. Available: https://examradar.com/cyber-security-mcq-set-1/

[84] F. R. Shamil. Web security and forensics multiple choice questions. [Online]. Available: https://t4tutorials.com/web-security-and-forensics-multiple-choice-questions/

[85] S. M. Abdullah and G. Vranken. Denial of service (DoS) in django. [Online]. Available: https://snyk.io/vuln/SNYK-PYTHON-DJANGO-456542

[86] Privilege escalation in django. [Online]. Available: https://www.cybersecurity-help.cz/vdb/SB2019120805

[87] A. Ravindran. Understanding django channels. [Online]. Available: https://arunrocks.com/understanding-django-channels/

[88] A. Mishra1. Demystifying django's ASGI. [Online]. Available: https://community-z.com/communities/tectoniques/articles/554

[89] readthedocs. Channel layers — channels 3.0.4 documentation. [Online]. Available: https://channels.readthedocs.io/en/stable/topics/channel_layers.html#single-channels

[90] ——. Django cryptography documentation. [Online]. Available: https://django-cryptography.readthedocs.io/en/latest/settings.html

[91] D. documentation. Password management in django. [Online]. Available: https://docs.djangoproject.com/en/3.2/topics/auth/passwords/

[92] Wikipedia. PBKDF2. Page Version ID: 1037722647. [Online]. Available: https://en.wikipedia.org/w/index.php?title=PBKDF2&oldid=1037722647

[93] MIcrosoft. Microsoft open source software security. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/opensource

[94] OpenJDK. [Online]. Available: https://openjdk.java.net/

[95] Rapid7. What is a web application firewall (WAF)? explained. [Online]. Available: https://www.rapid7.com/fundamentals/web-application-firewalls/

[96] ciberseguridad.com. Plan de respuesta a incidentes de seguridad. [Online]. Available: https://ciberseguridad.com/normativa/espana/medidas/plan-respuesta-incidentes-seguridad/

[97] Wikipedia. Business continuity planning. Page Version ID: 1040693242. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Business_continuity_planning&oldid=1040693242

[98] T. Habets. Google authenticator OpenSource. Original-date: 2014-10-08T17:54:27Z. [Online]. Available: https://github.com/google/google-authenticator

[99] D. G. Team. What is MIME sniffing? - KeyCDN support. [Online]. Available: https://www.keycdn.com/support/what-is-mime-sniffing

[100] C. Hope, "When was the first computer invented?" [Online]. Available: https://www.computerhope.com/issues/ch000984.htm

[101] G. Jevtik, "What is SDLC? How the Software Development Life Cycle Works," May 2019. [Online]. Available: https://phoenixnap.com/blog/software-development-life-cycle

[102] A. Miller, "Secure SDLC | Secure Software Development Life Cycle | Snyk," 2020. [Online]. Available: https://snyk.io/learn/secure-sdlc/

[103] G. Erdogan. (2009) Security testing of web based applications. [Online]. Available: https://www.researchgate.net/publication/268418932_Security_Testing_of_Web_Based_Applications

[104] Y. Nader. Top 10 open source security testing tools for web applications (updated). [Online]. Available: https://hackr.io/blog/top-10-open-source-security-testing-tools-for-web-applications

[105] DjangoProject. The web framework for perfectionists with deadlines | django. [Online]. Available: https://www.djangoproject.com/

[106] Wikipedia. WebSocket. Page Version ID: 1036283258. [Online]. Available: https://en.wikipedia.org/w/index.php?title=WebSocket&oldid=1036283258

[107] O. S. Hiremath. Top 50 cybersecurity interview questions | cybersecurity training. Section: Cyber Security. [On-

line]. Available: https://www.edureka.co/blog/interview-questions/cybersecurity-interview-questions/

[108] readthedocs. ASGI — channels 3.0.4 documentation. [Online]. Available: https://channels.readthedocs.io/en/stable/asgi.html