

DEGREE: Degree in Computer Engineering
Management and Information Systems

FINAL DEGREE PROJECT

Online penetration testing laboratory

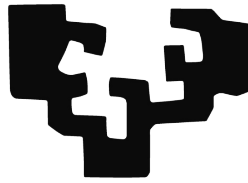
Student: Abad Garcia, Gorka

Director (1): Pereira Varela, Juan Antonio

Course 2018-2019

Date: Bilbao, June 16, 2019

eman ta zabal zazu



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

ENGINEERING-DEGREE IN COMPUTER MANAGEMENT
AND INFORMATIONS SYSTEMS

FINAL DEGREE PROJECT

Online penetration testing laboratory

Gorka Abad Garcia

directed by
Juan Antonio PEREIRA VARELA

Bilbao, June 23, 2019

Abstract

This work introduces a web application that helps to learn and to improve users' computer security related skills. For that aim is proposed to build an online virtual laboratory with some online containerized applications. If virtual machines had been used, instead of containers, computers with less power would have not been able to run them. Those apps are composed in a way that allows users' to skip the tough part of the configuration. Containers will be hosted online so they can be reached from anywhere and anyone at any time.

Once the environment is set up the next step is to gain the most cybersecurity skill or knowledge as possible. To accomplish this, a nice looking web interface simplifies the task of learning. Using a Capture the Flag (CTF) like scheme users' will gain theoretical and practical knowledge in the field of security in systems. In order to maintain consumers engaged, challenges have different levels of difficulty which are categorized from 1 to 5. Consumers will get points when completing challenges, and they will be appearing in a leaderboard, creating an environment of competitiveness that builds a link between learning and challenge.

Keywords: Computer security, CTF, Docker, Container, Cloud computing, Web laboratory

Resumen

Este trabajo constituye una importante ayuda para el aprendizaje en el ámbito de la seguridad informática. Trata de mejorar las habilidades de los usuarios mediante retos que deben superar. Para ello se ha diseñado una aplicación web que ayuda al aprendizaje. Los retos a superar están encapsulados en pequeños contenedores que contienen lo necesario para el reto en concreto.

Muchas aplicaciones web similares preparan los retos para ser ejecutados en el ordenador del usuario por medio de una virtualización del entorno, lo que conlleva una demanda de potencia.

Además el entorno debe ser configurado manualmente. Puede que los usuarios no tengan ni la potencia ni los conocimientos suficientes para poder preparar el entorno. Para ello y a modo de solución ha sido diseñado este proyecto, que con el uso de entornos o retos preconfigurados en la nube, cualquier usuario puede acceder a ellos sin importar la potencia de su ordenador, y sin tener que realizar ninguna configuración.

Además a modo de hacer más entretenido el aprendizaje se proponen retos a modo de Obtener la bandera (Capture the flag CTF). Los usuarios deberán obtener la bandera superando retos y obtendrán puntos por ello. Los mejores usuarios aparecerán en un ranking, aumentando así su afán por mejorar.

Palabras clave: Seguridad informática, CTF, Docker, Contenedores, Computación en la nube, Laboratorio web

Laburpena

Proiektu hau sistema informatikoen segurtasunaren ikaskuntzara gidatzen da. Erabilzaileen gaitasunak hobetzea dauka helburu. Horretarako aplikazio web bat garatu egin da, ikaskuntza errazteko. Hainbat erronka prestatu dira, informatikako segurtasunari gidatuak, erabiltzaileek gainditu behar dituztenak, gainditzen badituzte puntuak lortuko dituzte, eta hoberenak ranking batean agertuko dira. Proiektu hau proposatzen dituen erronkak Bandera Harrapatzea (Capture the Flag CTF) bezalakoak dira, non erabiltzaileek gako batzuk lortu behar dituzte, erronkak gainditzeko.

Hainbat antzeko aplikazio daude, baina ia guztiak ez dituzte ikaskuntzarako ingurumena, adibidez erronkak, prestatzen edo konfiguratzeko, erabiltzaileek prestatu behar dituzte, batzuetan zaila izan dezake. Ingurumenak jaisten dira, gainera ordenagailuek nahiko ahaltsuak izan behar dira softwarea erabiltzeko.

Arazo hau konpontzeko aplikazio jada prestatuak eta deskargatu gabekoak sortu egin dira. Erronkak kontainerizatu egin dira, horrela erabiltzaileek ez dituzte ezer ez prestatu behar ezta deskargatu ere. Kontainerrak erronka gauzatzeko behar den ezinbestekoa dauka bere barnean, honek oso arinak egin ditu kontainerrak. Kontainer hauek hodeian daude eta erraz koduetau egin daitezke. Beraz ordenagilueen ahalmena ez da mugapen bat, edonork saia ditzake, ordengailuaren ahalmena kontuan hartu gabe, eta erronkak prestatu gabe.

Hitz gakoak: Seguratu informatikoa, CTF, Docker, Kontainerrak, Konputazioa hodeian, Web laborategia

Contents

| | |
|--|-----------|
| 1. Terms | 11 |
| 2. Introduction | 12 |
| 3. Initial approach | 17 |
| 3.1. Used tools | 17 |
| 3.2. Scope | 18 |
| 3.2.1. Management | 20 |
| 3.2.2. Learning | 21 |
| 3.2.3. Analysis and design | 21 |
| 3.2.4. Testing | 22 |
| 3.2.5. Implementation | 23 |
| 3.2.6. Documentation | 24 |
| 3.3. Time planning | 25 |
| 3.4. Economic Analysis | 27 |
| 3.4.1. Risk analysis | 28 |
| 4. Defining requirements | 30 |
| 5. Tool selection | 32 |
| 5.1. Front end | 33 |
| 6. Architecture | 34 |
| 6.1. Validating users' credentials | 34 |
| 6.2. Interacting with the database | 35 |
| 6.3. Containerisation | 35 |
| 6.4. Securely storing passwords | 36 |
| 7. Development | 38 |
| 7.1. Challenges | 42 |
| 7.2. Server | 46 |
| 8. Testing | 47 |
| 9. Future work | 49 |
| 10. Conclusions | 50 |
| 10.1. Goals | 50 |
| 10.2. Time scheduling | 50 |

| | |
|---|-----------|
| A. Installation guide | 53 |
| A.1. Prerequisites | 53 |
| A.2. Installing the web application | 54 |
| B. User guide | 56 |
| B.1. Login and Sign Up | 56 |
| B.2. Main page | 57 |
| B.3. Challenge | 58 |
| B.4. Leaderboard | 59 |
| B.5. Settings | 59 |
| A. Testing guide | 61 |
| A.1. Codeception installation and configuration | 61 |

List of Figures

| | | |
|-----|---|----|
| 1. | Diagram of the actual internet. | 12 |
| 2. | Most common password used. | 13 |
| 3. | OWASP 10 most common vulnerabilities. | 14 |
| 4. | Learning diagram. | 15 |
| 5. | WBS diagram. | 19 |
| 6. | GANTT diagram. | 26 |
| 7. | Virtual machines and Docker comparison. | 32 |
| 8. | Database schema. | 36 |
| 9. | Password encryption. | 37 |
| 10. | Architecture diagram. | 37 |
| 11. | Password recovery diagram. | 39 |
| 12. | Searches about docker along the time. | 40 |
| 13. | Vigenere square. | 43 |
| 14. | SQL Injection use nowadays. | 44 |
| 15. | Successfully testing. | 47 |
| 16. | Failed testing. | 48 |
| 17. | Login page. | 54 |
| 18. | Sign up page. | 56 |
| 19. | Password recovery page. | 56 |
| 20. | Main page. | 57 |
| 21. | Tear down menu. | 58 |
| 22. | Challenge. | 59 |
| 23. | Leaderboard. | 59 |
| 24. | Settings. | 60 |

List of Tables

| | | |
|----|---|----|
| 1. | Time estimation | 25 |
| 2. | Total cost of the project | 28 |
| 3. | Time estimation and invested time | 51 |

1. Terms

- CTF: Capture the flag
- XSS: Cross site scripting
- IoT: Internet of things.
- WPA2: Wi-Fi Protected Access II.
- Virtual Box: Virtualization tool.
- VMWare: Virtualization tool.

2. Introduction

A lot has changed since the Internet appeared, as a way to connect computers and transfer data, to nowadays where the Internet is used everywhere every time. Connections over the internet are achieved by sending and receiving packets, those packets are sent over a protocol. Designed 40 years ago, the TCP/IP protocol was developed, along with a complex way of interconnecting computers. Nowadays the internet has changed a lot and it has become more complex as it can be seen in 1 ¹.lot

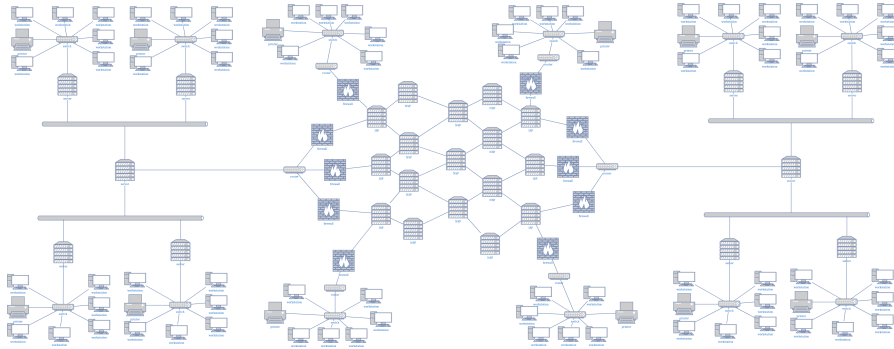


Figure 1: Diagram of the actual internet.

TCP/IP protocol was not safe, the Internet was not safe, it was not developed to be a safe tool. No one would expect that the Internet would be used in everything, like mobile phones, televisions, coffee machines, fridges or even in cardiac pacemakers. Vulnerabilities are found almost every day. Therefore, computers or IoT(Internet of Things) devices are not safe in itself. Often patches are released in order to fix the vulnerability but sometimes the responsibility to apply them is from the final user who may not know how to. In recent years, all kinds of vulnerabilities have been found i.e: Vulnerabilities in WPA/WPA2, SQL injection, XSS. Also, there are some bad practices that allow an attacker to gain control of a computer i.e.: the use of simple passwords ², outdated software or use of untrusted software.

¹Image taken from Draw.io

²Image taken from Mozilla internethealthreport.org

| | | | | |
|---------------------|---------------------|-----------------------|---------------------|--------------------|
| 1. 123456 | 11. 123123 | 21. mustang | 31. 7777777 | 41. harley |
| 2. password | 12. baseball | 22. 666666 | 32. f*cky*u | 42. zxcvbnm |
| 3. 12345678 | 13. abc123 | 23. qwertyuiop | 33. qazwsx | 43. asdfgh |
| 4. qwerty | 14. football | 24. 123321 | 34. jordan | 44. buster |
| 5. 123456789 | 15. monkey | 25. 1234...890 | 35. jennifer | 45. andrew |
| 6. 12345 | 16. letmein | 26. p*s*y | 36. 123qwe | 46. batman |
| 7. 1234 | 17. shadow | 27. superman | 37. 121212 | 47. soccer |
| 8. 111111 | 18. master | 28. 270 | 38. killer | 48. tigger |
| 9. 1234567 | 19. 696969 | 29. 654321 | 39. trustno1 | 49. charlie |
| 10. dragon | 20. michael | 30. 1qaz2wsx | 40. hunter | 50. robert |

Figure 2: Most common password used.

Even though easy fixes are well known, users tend to not apply them immediately, so this scenario leads to a huge problem. If their data is not properly protected it can lead to data leaks where cybercriminals steal victims data. Criminals use techniques of scamming, blackmailing or phishing. People do not realize how much data of them is on the Internet, and even less the worth of it.

So as the world has become more digitized the more vulnerabilities appear. Even though some of them are being fixed as they appear, the inner techniques used by those vulnerabilities still remain for future uses with different approaches. For vulnerability exploiting, practices or scopes may vary but in essence, the problem persists. Those are the 10 most common vulnerabilities ³. Once a brief introduction to cybersecurity is done, how can it be learned in order to make the Internet a safer place for the people?

³Image taken from owasp.org

| OWASP Top 10 - 2013 | → | OWASP Top 10 - 2017 |
|--|---|--|
| A1 – Injection | → | A1:2017-Injection |
| A2 – Broken Authentication and Session Management | → | A2:2017-Broken Authentication |
| A3 – Cross-Site Scripting (XSS) | ↘ | A3:2017-Sensitive Data Exposure |
| A4 – Insecure Direct Object References [Merged+A7] | U | A4:2017-XML External Entities (XXE) [NEW] |
| A5 – Security Misconfiguration | ↘ | A5:2017-Broken Access Control [Merged] |
| A6 – Sensitive Data Exposure | ↗ | A6:2017-Security Misconfiguration |
| A7 – Missing Function Level Access Contr [Merged+A4] | U | A7:2017-Cross-Site Scripting (XSS) |
| A8 – Cross-Site Request Forgery (CSRF) | ☒ | A8:2017-Insecure Deserialization [NEW, Community] |
| A9 – Using Components with Known Vulnerabilities | → | A9:2017-Using Components with Known Vulnerabilities |
| A10 – Unvalidated Redirects and Forwards | ☒ | A10:2017-Insufficient Logging&Monitoring [NEW,Comm.] |

Figure 3: OWASP 10 most common vulnerabilities.

When cybersecurity comes into play, a wide knowledge is needed, almost a lifetime of experience is not enough to become the greatest professional. As in every field, no matter if it is learning football or learning a new language, the beginnings are not easy. The journey of learning begins sinking in the immense world of internet fulfilled with documentation, everything looks tough, too much information without a clear start point as shown at 4⁴.

⁴Image taken from Dev.to

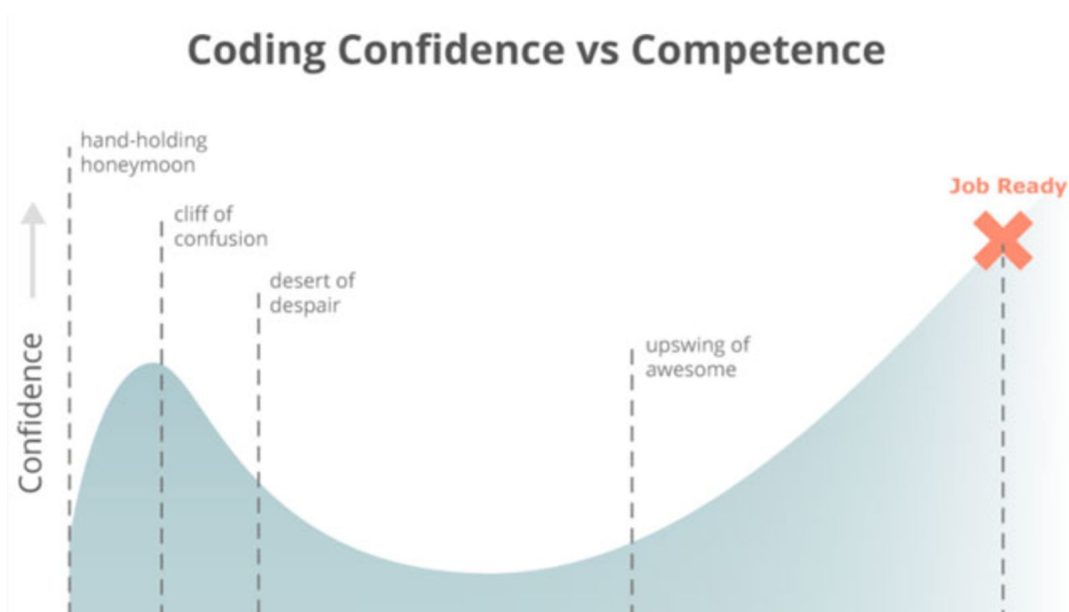


Figure 4: Learning diagram.

As is shown above at 2 the very first moments of learning are full of excitement, but unfortunately, the reality hits hard. In order to simplify the task of dealing with hundreds of starting points, a wide variety of websites have been made. In the specific field of cybersecurity is worth of mentioning these ones: Pentester lab, Pentester box, or Hacker one.

The last one provides great tutorials with different levels of complexity. And the other two, are more likely to what this project is aiming to. All of them are great but, they do not provide an online environment to make penetration testing easy or straightforward, at least at the beginnings. However, they provide vulnerable images that the user can download and use them locally. This can be an impossible task for users that have not enough powerful computer to run several virtual machines o even a single one. Furthermore, setting up those machines is not an easy task. Firstly, the user has to download the image, then set up the software to recreate the machine (tools like Virtual Box or VMware), later the image has to be installed on the virtual machine. And at last, and the most difficult part, configure the virtual machines. This involves: setting-up network adapters, choosing a network type(NAT, Bridged, NAT Network...), port forwarding and several other options have to be considered.

On that setting, it is clear that learning is not an easy task, and even less when the user has to configure everything on his own, thus CTF(Capture The Flag) like games are used to learn. CTF is a game in which the goal is to get a flag. A flag can be anything, from a text file to a password or an image. This game like a challenge is great because the users achieve points when they complete a task when they find the flag. It encourages them to keep trying new challenges. There is a wide range of categories i.e.: Web, Forensic, Crypto, Binary.

- Forensic challenges are those how can include file format analysis, steganography, memory dump analysis, or network packet capture analysis.
- In crypto challenges, the main goal is usually to crack a cryptography algorithm in order to get a flag.
- In Binary challenges usually are needed to find a vulnerability in a program, commonly Linux ELF files or Windows EXE files.

Sometimes a task has a predecessor and a successor, in order to get into the next challenge the previous one has to be completed. Some CTFs examples could be: Hacker One CTF, INCIBE CTF, and maybe the most important one DEFCON CTF. Therefore, this methodology sinks the users into a challenging environment in which learning and having fun are guaranteed. Furthermore, there are competitions where several teams challenge between them, there are also attack-defense competitions, every single of them are perfect scenarios to demonstrate their own skills and of course learn.

Due to the benefits CTF has, some of the inner essences have been taken to design this project, users will learn while they challenge themselves in a CTF like environment, earning points and sharing results with the community while every challenge is configured and online. Regardless of which computer the user has or where the user is, the challenges are accessible all over the world ready to use, in a user-friendly web environment.

Also, the project is aimed to use in lessons where there are a considerable number of students. Setting up every single computer in order to try out what has been explained can be a really tough task. So this online web laboratory fits perfectly in this context.

3. Initial approach

3.1. Used tools

This section describes all the tools needed to develop the project.

- **GitHub:** This tool is used for code version control. Used for ensuring the last version of the project is used, furthermore, the older version can be checked in the case is needed.
- **PHP:** A server-side scripting language designed for Web development.
- **MySQL:** A open-source relational database management system. It is used for storing data, such as user information.
- **HTML:** Is the standard markup language for creating web pages and web applications.
- **CSS:**A style sheet language used for describing the presentation of a document written in HTML (in this project).
- **JavaScript:** Is a programming language, which allows making interactive web applications.
- **Docker:** An open source platform for developers and sysadmins to develop, ship, and run applications.
- **Visual Studio:** An IDE for developing apps.
- **MySQL Workbench:** An application that helps to manage databases.
- **Overleaf:** An online LaTeX editor.
- **Bootstrap:** A framework that helps developing web applications.
- **Google Cloud:** A cloud services platform.
- **Gantt Project:** An application that helps to manage tasks and creating Gantt's diagram.
- **Cacoo:** A tool used to create diagrams or other graphics.
- **Overleaf:** Is an online LaTeX and Rich Text collaborative writing and publishing tool that makes the whole process of writing, editing and publishing scientific documents much quicker and easier.
- **Codeception:** A automated testing framework base on PHP Unit.

3.2. Scope

For developing this project a WBS diagram was made, as can be seen in figure 5. The diagram is divided into 6 big clusters that are sub-divided into other groups.

- **Management:** This group is needed for tracking the project.
- **Learning:** Here appears everything that has to be learned to make this project.
- **Analysis and design:** Project functionalities and how to make their implementation are written down.
- **Implementation:** The implementation needed to build all the functionalities of the project.
- **Test:** This block is for ensuring that everything works well in the project if is not, it has to be fixed.
- **Documentation:** The documentation of the project that has to be done along with the project duration.

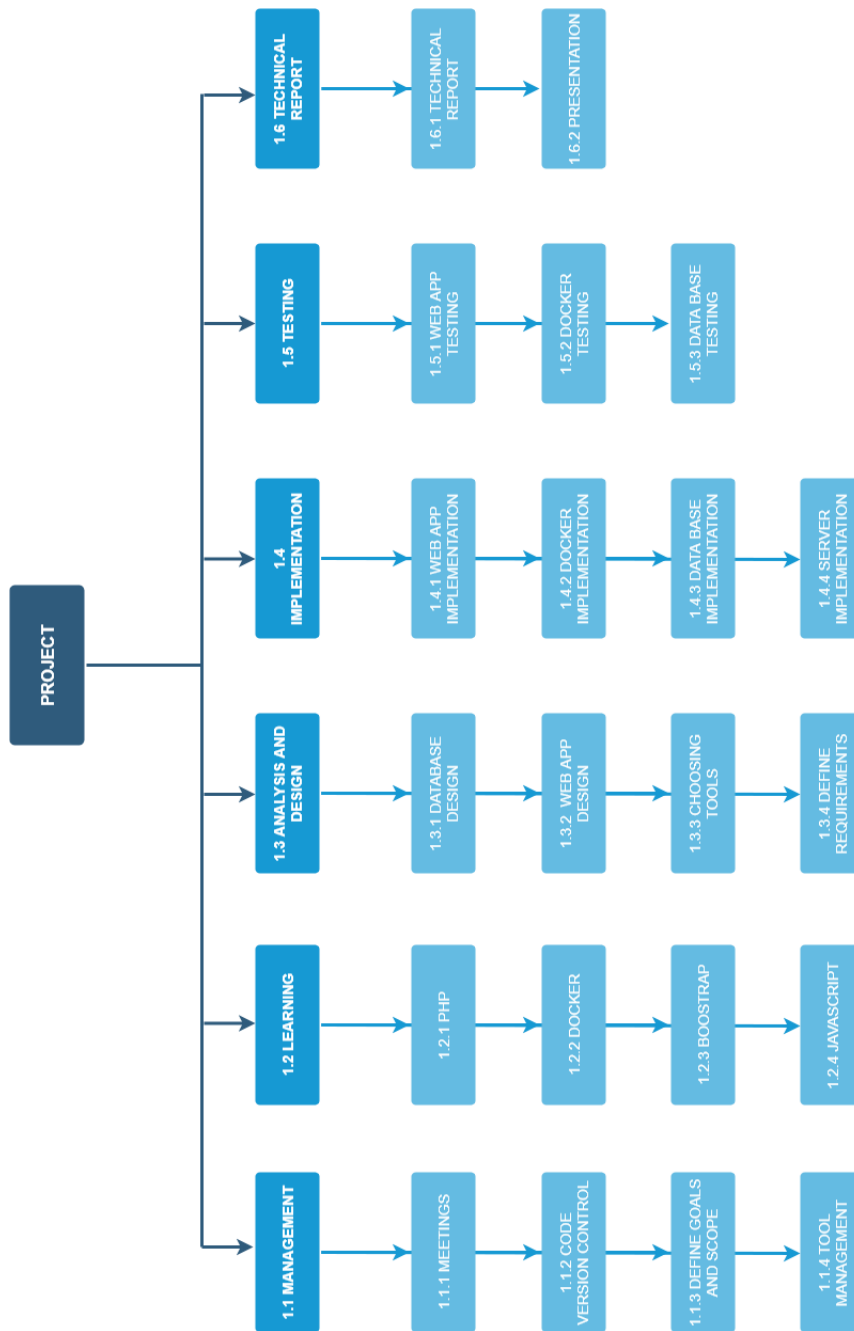


Figure 5: WBS diagram.

3.2.1. Management

Meetings

- Time estimation: 7 hours.
- Description: Meetings done with the director of the project along its development.
- Output: Decisions made on meetings.
- Resources: A computer.

Setting up code version control software

- Time estimation: 10 hours.
- Description: Configuring Git Hub repository and upload/download files along the project.
- Output: Nothing.
- Resources: A computer.

Define goals and scope

- Time estimation: 4 hours.
- Description: Define the project goals list all the to dos.
- Output: A list with the to dos.
- Resources: A computer.

Tools management

- Time estimation: 10 hours.
- Description: Select all the tools that will be used, comparing different ones and choosing the best ones.
- Output: A list of every tool used.
- Resources: A computer.

3.2.2. Learning

PHP

- Time estimation: 25 hours.
- Description: Learning everything needed about PHP to develop this project.
- Output: Nothing.
- Resources: A computer, PHP books and guides.

Docker

- Time estimation: 25 hours.
- Description: Learning about containers and its functionalities, especially about Docker and Docker Compose.
- Output: Nothing.
- Resources: A computer, Docker documentation and guides.

Bootstrap

- Time estimation: 25 hours.
- Description: Learning about front end developing using Bootstrap.
- Output: Nothing.
- Resources: A computer, Bootstrap documentation and guides.

Java Script

- Time estimation: 25 hours.
- Description: Learning basic DOM manipulation using Java Script.
- Output: Nothing.
- Resources: A computer and Mozilla documentation.

3.2.3. Analysis and design

Database:

- Time estimation: 5 hours.
- Description: Define the database model to fit with the project specifications.
- Output: A database model.
- Resources: A computer and MySQL workbench.

Web Application

- Time estimation: 12 hours.
- Description: Design the look and decide the architecture of the web application.
- Output: A well structured web application.
- Resources: A computer.

Choose tools

- Time estimation: 2 hours.
- Description: Define the tools that will be used.
- Output: A list of tools.
- Resources: A computer.

Defining Requirements

- Time estimation: 5 hours.
- Description: Define the functionalities of the project.
- Output: A list of functionalities.
- Resources: A computer.

3.2.4. Testing

Web application:

- Time estimation: 5 hours.
- Description: Test the entire web application.
- Output: Nothing.
- Resources: A computer and the web application finished.

Docker containers

- Time estimation: 2 hours.
- Description: Test the Docker containers.
- Output: Nothing.
- Resources: A computer and the Docker containers finished.

Database

- Time estimation: 1 hours.
- Description: Test the database.
- Output: Nothing.
- Resources: A computer.

3.2.5. Implementation

Web application:

- Time estimation: 125 hours.
- Description: Develop the web application using JS, PHP and HTML, and creating connections to database from the back end.
- Output: A developed web application.
- Resources: A computer and needed software for the development.

Docker containers

- Time estimation: 100 hours.
- Description: Code the Docker containers in order to be used in the web application according to the challenge.
- Output: Docker containers created.
- Resources: A computer, Docker and Docker Hub.

Data Base

- Time estimation: 5 hours.
- Description: Once designed, make the implementation of the database.
- Output: An usable database.
- Resources: A computer and MySQL.

Server

- Time estimation: 25 hours.
- Description: Configure and prepare the server to deploy the web application.
- Output: An operative server.
- Resources: A computer and a server.

3.2.6. Documentation

Technical Report:

- Time estimation: 75 hours.
- Description: Write down the document that explains the entire project.
- Output: Nothing.
- Resources: A computer and overleaf.

Presentation

- Time estimation: 15 hours.
- Description: Prepare the presentation for defending the thesis.
- Output: Nothing.
- Resources: A computer.

3.3. Time planning

Table 1: Time estimation

| Activities | Estimation (in hours) |
|------------------------|------------------------------|
| Management | 29 |
| Meetings | 7 |
| Code version control | 10 |
| Define goals and scope | 2 |
| Tools management | 10 |
| Learning | 100 |
| PHP | 25 |
| Docker | 25 |
| Bootstrap | 25 |
| Java Script | 25 |
| Analysis and design | 24 |
| Data base design | 5 |
| Web application design | 12 |
| Choosing tools | 2 |
| Defining requirements | 5 |
| Implementation | 255 |
| Web application | 125 |
| Docker containers | 100 |
| Data base | 5 |
| Server | 25 |
| Test | 8 |
| Web application | 5 |
| Docker containers | 2 |
| Database | 1 |
| Documentation | 90 |
| Technical Report | 75 |
| Presentation | 15 |
| Total | 506 |

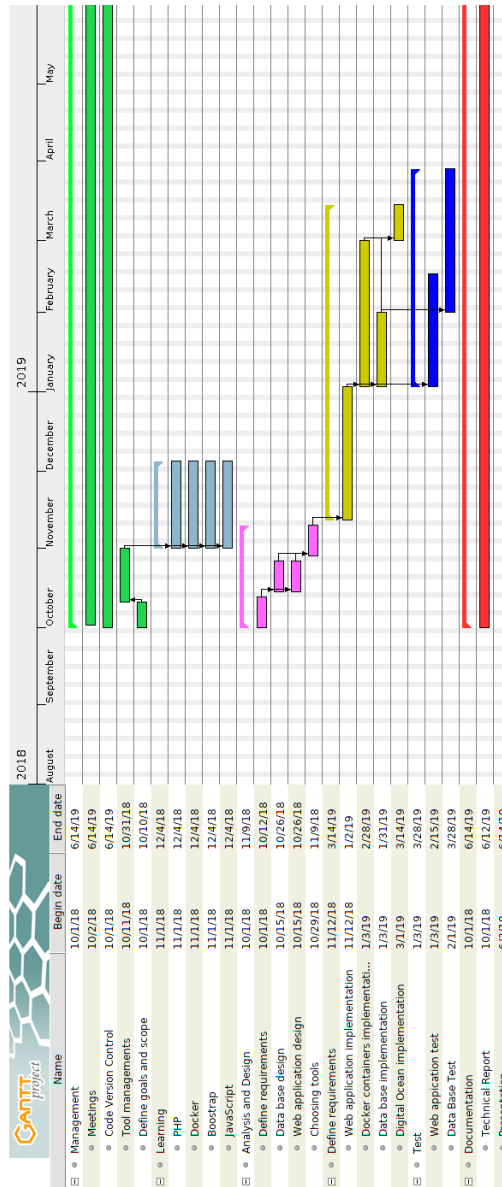


Figure 6: GANTT diagram.

3.4. Economic Analysis

This is a project for the university, so any kind of economic retribution is not expected. However, in this section, an economic analysis will be done, to estimate the costs.

Manpower

According to the BOE (Sec. III. Page 4373) the average salary for a programmer is 1.208,40€per Month, that is a total of 16.917,60€per year, without any kind of extra salary as incentives. Salary is calculated using this formula:

$$Salary/hour = \frac{Monthly\ Salary}{Week/Month * Work\ Days/Week * Work\ Hours/Day} \quad (1)$$

Often, every month has 4 weeks, and each week has 5 workdays of 8 hours. According to the equation, this would be the output:

$$Salary/hour = \frac{1208.40}{4 * 5 * 8} = \frac{1208.40}{160} = 7.5525 \quad (2)$$

It makes the total amount of 7.5525€per hour.

To make this project the estimated time is 372 hours:

$$Salary = Work\ Hours * Salary/Hour = 372 * 7.5525 = 2809,53 \quad (3)$$

At last, 2809,53€of manpower is needed for the whole project.

Software

Different software has been used. Even though most of them are free, there is some like PHP Storm which is free for academic purposes. Also, in the scope of cloud computing, the instance used in Google Cloud was free because it was under an academic license.

Hardware

In the hardware section, only a laptop has been needed. The average lifetime of a laptop is about 4 years. The laptop used cost around 350€.

$$Monthlycost = \frac{laptop\ price}{4\ years} = \frac{350}{12 * 4} = 7,3e/month \quad (4)$$

The monthly cost of the laptop is 7.3€if the project has been developed in 8 months, then the resultant cost of the hardware is 58.3€.

Table 2: Total cost of the project

| <i>Type of spending</i> | <i>Spending (€)</i> |
|-------------------------|---------------------|
| Manpower | € 2809,53 |
| Software | € 0 |
| Hardware | € 58.3 |
| Total | € 2,867.83 |

Total

In this table are grouped all the costs of the project. After every calculation as show in the 2 the total cost is 2,867.83€.

3.4.1. Risk analysis

In a project development can occur some risks, this contingency plan has been designed to identify those risks and try to avoid them. Moreover, if they occur, having a plan to restore them to the previous state is a must.

Problems whit the computer

- Description: The computer suddenly shuts down losing partial or total information stored on it. Or simply a memory issue happens to lose some of the data.
- Probability: Medium
- Impact: High
- Prevention: Keeping a duplicate of the data on the internet, on a cloud service for example.
- Contingency plan: Keep working on another computer with the data restored from the security copy online.

Become ill

- Description: Due to any kind of illness being unable to work.
- Probability: Medium
- Impact: High
- Prevention: Ensure good health and visiting the doctor at any kind of rare behavior.
- Contingency plan: Follow the instructions provided by the doctor.

Bad time estimation

- Description: Due to a miscalculation on the time schedule, not being able to finish the project for the deadline.
- Probability: Low
- Impact: High
- Prevention: Estimate more time than expected.
- Contingency plan: Postpone the deadline or omit some optional features.

Thirty part services issue

- Description: Because of any kind of issue of thirty parts software used on the project, having a miss behavior on the project normal workflow.
- Probability: Low
- Impact: Medium
- Prevention: Keep tracking of the thirty-part software status and any kind of notice.
- Contingency plan: Use an outdated thirty part software temporally.

4. Defining requirements

In this section minimum requirements for the development of the project are going to be listed and explained.

- **Challenges:** As one of the main goals of the project was to make a capture the flag like environment to learn cybersecurity-related skills. This is made by using challenges. Challenges are tasks that users have to completely relate to computer security, they have to find a flag, an objective such as passwords or text files. If they complete the challenge they will earn points according to its difficulty. There are 5 types of difficulty, being 1 the easiest and 5 the hardest. This type of challenge allows the developer to create more challenges on the fly, creating like this a wide range of challenges for every kind of user.
- **Users:** In order to make the project personal and sociable, every user has its own profile. Every profile has information about completed challenges and punctuation. Every user can check other users' profiles. Best users will appear in a leaderboard.
- **Leaderboard:** The leaderboard is where the best users appear, the more points the higher they will be. This type of social ranking encourages users to try harder and competition between them.
- **Points:** Every user will earn points for a completed challenge. Depending on the difficulty of the challenge, the number of points earned may vary.
- **Log in and Sign up:** In order to get access to the user to her/his profile, a login is needed if an account is already created. If not, a profile can be created using the sign-up.
- **Password Recovery:** The password is securely stored in the database. Password is encrypted in a one-way direction, which means that they can not be recovered. A fix to this is to create a token that is sent via email to the user in order to reset the password with a new one. Using this method they are ensured that only the owner of the account will be able to reset its password. How passwords are encrypted will be explained in the Architecture section.
- **General:** Eventually what is going to be listed below is not particular for the coding of this project, it is related to the project in general.
 - **Modularity:** Because of the structure and architecture, this project has it can lead to an expansion or an improvement of it in an easy way. The project can be increased by adding, for example, more challenges to solve or adding a new look to the front-end.
 - **Learning:** Increasing my knowledge upon new technologies is a personal objective, this project led me to an insane learning challenge. As is explained below for building up this project lot of research has been made, along with

this research new methodology and new tools been appearing which were new for me. Moreover, another kind of knowledge such as planning or resolving problems on the fly has been acquired too.

5. Tool selection

Due to the dimension of this project, it was compulsory to provide a version control system in order to have a backup and a controlled version of the software. To accomplish this task, GitHub was chosen. It is one of the most commonly used distributed web-based version control systems, and it fits perfectly for this project.

Additionally, for the development of the actual code, PHP Storm was chosen, a well-suited IDE for PHP development, as long as PHP is what needs to be developed. In the same way, PHP storm can be used for the development of applications using other programming languages, such as JavaScript or HTML. Conversely, there are perfects IDE for developing software out there, great Open Source projects (i.e. Atom, VSCode) that can accomplish this take as good as PHP Storm does.

Typically when referring to penetration testing, with learning purposes, vulnerable machines or applications have to be created or replicated in order to try to come up with a vulnerability that can be exploitable. Usually, virtual machines are created to deal with those vulnerable versions. However, it can be quite useless, since virtual machines require a heavy load of RAM memory along with fast CPUs and HDDs or SSDs. In order to solve that problem, containerized applications were made. [10] Docker is a container platform to cost-effectively build and manage applications. In other words, Docker has the ability to encapsulate (containerize) applications or OS, and run them, with a very low-cost ⁵. Essentially it is a great option for making vulnerable applications, it aids to run several containers at the same time at a little cost.

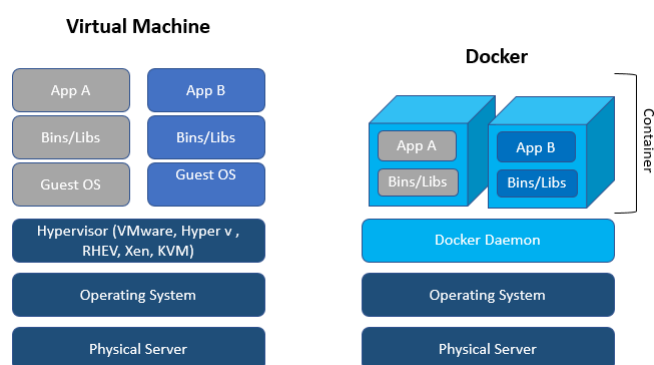


Figure 7: Virtual machines and Docker comparison.

As managing several containers at the same time could be a tough task, commonly Kubernetes is used along with Docker to solve that problem. [2] Kubernetes is a container orchestration system for automating application deployment. Clearly, it fits perfectly with Docker. Moreover, Kubernetes provides a web interface to easily monitor and manage Docker containers besides rolling updates, to manage versions and update containers on the fly. Although Kubernetes is not necessary for the development of this project, it

⁵Image taken from Docker web page.

could be a great option if the size of the project increments. Even though it could be a great addition due to lack of time it is not going to be implemented in this project. It is explained at 9.

5.1. Front end

A good looking web interface is needed for engaging new users, and to make navigation in the page straightforward. Thus Bootstrap is used. Bootstrap provides a collection of CSS and JS libraries, achieving a good looking interface. Also, the web page is responsive, which means that gets adapted to any kind of screen size, even to smartphones.

Additionally, a database is needed for storing users' and challenges data. MySQL was chosen along with MySQL Workbench which is a tool that helps to manage a database (or several databases). Also for managing the database in the server environment PHP My Admin was used, it is similar to MySQL Workbench.

6. Architecture

This section clarifies the structure of the project, providing a comprehensive explanation of the architecture. It is important to isolate clusters as much as possible, while they are detached, even a huge change in the project can be easily merged without causing any kind of problem to it. The project can be split into 3 blocks:

Firstly the front-end. This cluster covers everything related to the client side, for instance, HTML, CSS, and JavaScript. It is related to the web page look or view.

Secondly the block of the containers. Containers are running over Docker, every container is a challenge. Docker containers could be managed using Kubernetes as explained at 9.

Finally the back-end. This is the server side, where every cluster is connected to it in different ways. Its main functionalities are: validating users credentials, containerization, interact with the database.

6.1. Validating users' credentials

When a user logs in or when signs up, the data provided has to be checked. The users' email and password are searched in the database, if there is a coincidence, the user can log in. Those credentials such as the user name, password or the email are sent to the back end in order to check for unexpected input. Inputs are compulsory to be filtered, always. If not, it can lead to a Cross Site Scripting attack or a SQL injection for example. In this case, the user only will input data that will be executed in a SQL sentence. PDO (??) has to be used in order to securely execute a sentence in SQL, according to PHP documentation. PDO Prepare prepares a statement for execution and returns a statement object, which can be executed securely with PDO execute. As PDO is maintained by PHP is highly recommended to use it, they also recommended it in their documentation. [4]

```

1 //PDO prepare example
2
3 //Create the connection
4 $conexion = new mysqli($host_db , $user_db , $pass_db ,
5 $db_name);
6
7 //Securely prepare the statement
8 $stmt = $conexion->prepare("SELECT iduser FROM $db_name.
9 $tbl_name WHERE email = ? ;");
10
11 //Parameters binding
12 $stmt->bind_param("s" , $email);
13
14 //Execute the statement
15 $stmt->execute();
16
17 //Retrieve the results as a cursor
18 $result = $stmt->get_result();
19
20 //Iterate over the result
21 if ($result->num_rows > 0) {
22     while($row = $result->fetch_assoc()) {
23
24     }
25 }

```

6.2. Interacting with the database

The database in this project stores information about users and containers. When a user logs in or signs up or when information about a certain user is needed the users' table is accessed. Other pieces of information, like user ranking, is not stored; the leaderboard is automatically generated by retrieving the users' points quantity and printed in the screen by descending order. Moreover, for the visualization of the challenges, the "container" table has to be consulted. This table encapsulates all the information (name, description, number of points and challenge difficulty) required to properly show those challenges.

6.3. Containerisation

Containerisation is the lightweight alternative to the full machine virtualization used by tools like Virtual Box.

According to Docker documentation: A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

Docker can define and run multi-container Docker applications. For this purpose, Docker Compose is used. In accordance with Docker documentation:

In another scope, the database is the next block, where all the data is stored, users credentials and container information besides. The users' information is stored at the user table and the information about the challenges in the container table. Every user can have more than one challenge, and a challenge can be completed for more than one user. If a challenge has been completed by a user, it will be added to the userContainer table.

This is its schema 8:

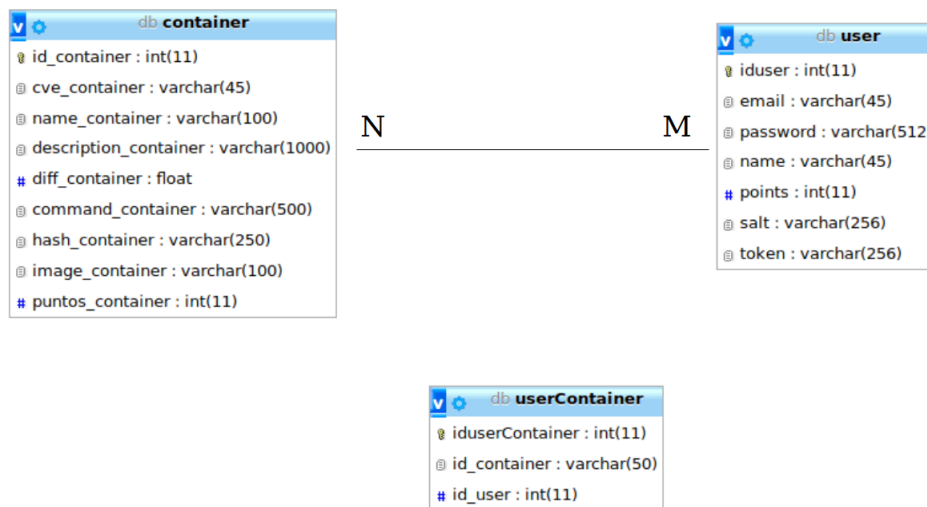


Figure 8: Database schema.

6.4. Securely storing passwords

Sensitive information such as passwords must be encrypted, ensuring data is not accessed by anyone. This prevents data from being revealed if the database is compromised. In this project, passwords have been encrypted using the same algorithm used by the Linux operating system to encrypt the user's passwords.

SHA 512 (Salt + Password)

Salt is random data that is used as an additional input to the users' password. The one-way hashing algorithm SHA 512 is applied. SHA 512 is a development of SHA 1 which is an MD4 based improvement. Original passwords cannot be decrypted from the hash, even using rainbow tables because of the salt, which randomizes the output.

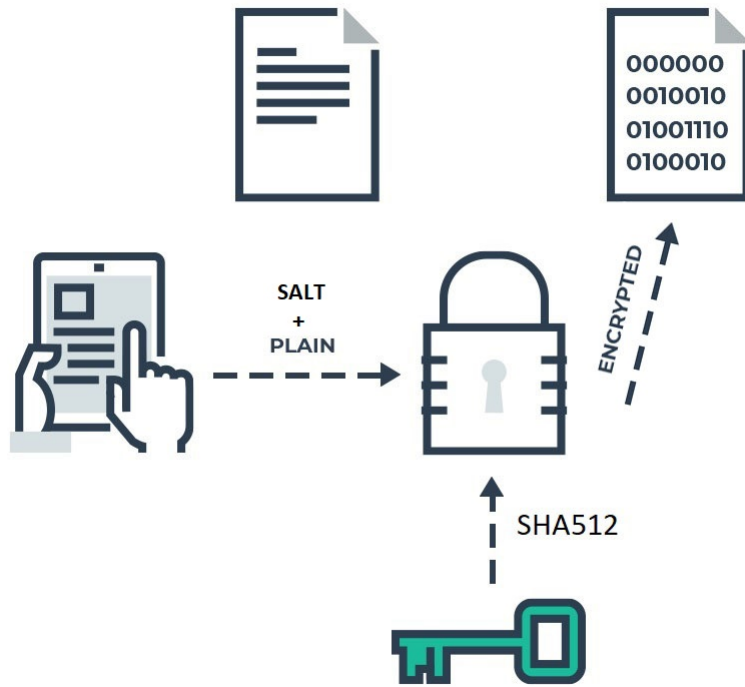


Figure 9: Password encryption.

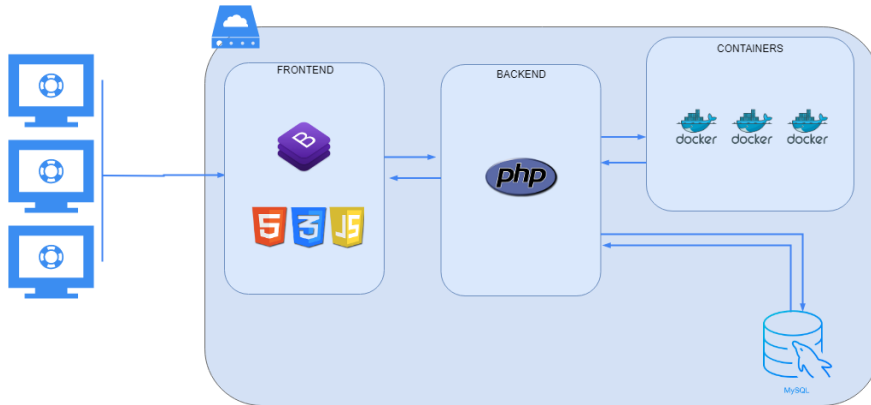


Figure 10: Architecture diagram.

7. Development

So far the architecture and the design along with the goal has been explained. Now in this section will be explained how the software was developed and how all the sections link to each other.

- **Web page look:** A web page style is developed using CSS. Bootstrap is a responsive, mobile first open source library for the front-end. It has great documentation, filled with educational examples that ease the way to make a good looking web page, even for novice designers.
- **Password recovery functionality:** As explained before because the hashing method used in passwords does not allow them from being recovered, because of the one-way algorithm. A solution to this, a password recovery method has been created. It consists of sending an email to the user with a link, along with some instructions, that redirects the user to a gateway. In that gateway, the password can be re-set. Below is explained comprehensively the method used to ensure users' passwords integrity.

Firstly the user has to write his email, after verifying that the email exists in the database an email is sent with a URL using Sendgrid (an email delivery service)⁶. At the same time, a token is created for the user with the provided email and it is saved in the database.

Secondly, the user must log in on his or her email account, and follow the instructions.

Finally, they will be redirected to a password recovery gateway, which has in the URL a token which was created previously, where they should type a new password and click on the reset button. After clicking the button the password is encrypted and updates the users' password only if then token matches.

⁶It is not recommended to send an email directly, using PHP's mail() function, due to the high odds that the message could be flagged as spam

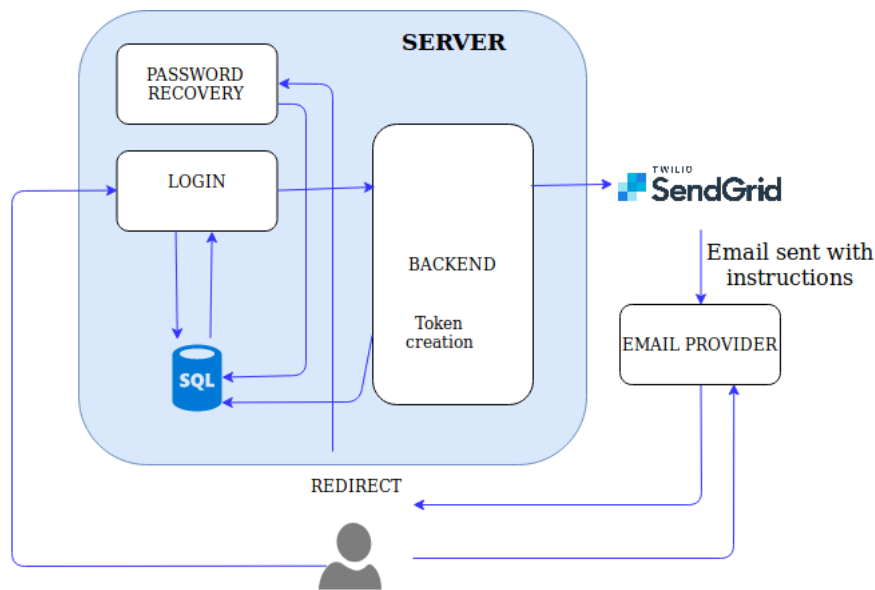


Figure 11: Password recovery diagram.

- **Containers:** As users will connect to an online machine in order to complete a challenge, those online machines could be running in as a virtual machine, at least this was our first thought. However, running several machines in a server is not a good practice because of two main points:
 - **Cost:** As virtual machines replicate a full OS, the computer which is hosting them will have a huge lack of CPU resources if the server is not enough powerful. Moreover, the server can be improved in order to get more power, but that will increase the server cost considerably. So that problem can be fixed using containers, which are tiny computers that run whatever is necessary to execute an app.
 - **Scalability:** In another scope using containers can improve systems' scalability, containers can be upgraded easily if a new version has been developed or an error has been found. Also adding new challenges is as easy as creating a container and uploading it. Docker provides a repository where containers can be uploaded, docker-hub. Other docker-hub users' containers can be downloaded too. Using this repository allows systems to download the latest version of the container if set, ensuring that software is up to date and working as intended.
- **Containers problems:** Containers have their own drawbacks, as follows:
 - **Learning curve:** Getting started with containers is not an easy task, despite there are guides or online courses the amount of time that has to be

invested may not be worth for some companies. [5] [3] [9] Only a few companies are built using containers, and re-build a company to fit into containers frame could be a tough task. Firstly because employees may not have the necessary knowledge to develop containers as containerization services are a relatively new technology so there could be a lack of experience in some scenarios. Moreover according to DZone could be 10% of network performance degradation.

- **Evolution:** As is a technology that is evolving, frequent updates can lead to security problems as the last one found CVE-2019-5736 that allows attackers to overwrite the host runc binary (and consequently obtain host root access) by leveraging the ability to execute a command as root. Or the Docker Hub data breach that exposed sensitive data from approximately 190.000 accounts, about 5% of Docker Hub community. Those vulnerabilities can lead companies to distrust of the containerization and make them wait until container companies consolidate more.



Figure 12: Searches about docker along the time.

As explained before, Docker containers are created by using Dockerfiles, and also could be created using docker compose which is used for running multi-container applications.

Using Compose is basically a three-step process:

- 1.- Define the app's environment with a Dockerfile so it can be reproduced anywhere.
- 2.- Define the services that make up the app in docker-compose.yml so they can be run together in an isolated environment.
- 3.- Run docker-compose up and Compose starts and runs the entire app.

In this project, Docker Compose has been used to recreate an SQL Injection environment.

This code below simply creates a PHP MySQLi environment.

```

1 Dockerfile
2
3 FROM php:7.2.2-apache # Base image
4

```

```
5 #Command to be executed once the image is created
6 RUN docker-php-ext-install mysqli
```

This code creates a multi-container application configuring and using local files previously created.

```
1 docker-compose.yml
2
3 version: "3.1" #Version
4 services:
5     www:
6         build: .
7         ports: #Port forwarding
8             - "8001:80"
9         volumes:
10            #Copying files from local to inside the container
11            - ./www:/var/www/html/
12        links:
13            - db
14        networks:
15            - default
16    db:
17        image: mysql:8.0 #Base image
18        ports: #Port forwarding
19            - "3306:3306"
20        command:
21            --default-authentication-plugin=mysql_native_password
22        environment:
23            MYSQL_DATABASE: myDb
24            MYSQL_USER: XXXX
25            MYSQL_PASSWORD: XXXX
26            MYSQL_ROOT_PASSWORD: XXXX
27        volumes:
28            #Copying files from local to inside the container
29            - ./dump:/docker-entrypoint-initdb.d
30            - ./conf:/etc/mysql/conf.d
31            - persistent:/var/lib/mysql
32        networks:
33            - default
34    phpmyadmin:
35        image: phpmyadmin/phpmyadmin #Base image
36        links:
37            - db:db
```

```

38     ports: #Port forwarding
39         - 8000:80
40     environment: #Credentials definition
41         MYSQL_USER: XXXX
42         MYSQL_PASSWORD: XXXX
43         MYSQL_ROOT_PASSWORD: XXXX
44 volumes:
45     persistent:

```

This is a part of the development needed for the SQL Injection challenge, detailed explained below at 7.1.

7.1. Challenges

In this section the challenges developed for this project are explained.

- Shell Shock:** The Shell shock vulnerability also known as Bashdoor is a kind of bug that allows the attacker to run arbitrary commands, permitting the attacker to gain unauthorized access to the system. [7] Commonly servers handle requests by using the Common Gateway Interface (CGI) which sends various details in the request, such as the user agent, which is used to identify the program who is sending the request. An attacker could send a bash script in the user agent and the server will execute it if is not properly sanitized. Bash scripts store functions as environment variables, unfortunately parsing of functions from environment variables can lead to unexpected effects. For example if after the function is added a command bash will execute it first, and then the function.

```

$ export foo='() { echo "Inside function" ; };
echo "This should not be executed"'

```

```

$ bash -c 'foo'
This should not be executed
Inside function

```

Note that whatever is after the function declaration can be malicious or not. This is an example with malicious code that can retrieve system passwords.

```

#An empty function and printing system passwords
() { ;; }; echo; echo; /bin/bash -c 'cat /etc/passwd'

```

Also it can be used using curl to get passwords from a remote server.

```

curl -H "user-agent: () { ;; }; /bin/bash -c 'cat /etc/passwd'"
\ http://<ip address>

```

```
#Server response
HTTP/1.0 200 OK
Server: nginx/1.2.1
Date: Sun, 19 May 2019 11:22:43 GMT
Content-Type: text/html
root:x:0:0:root:/root:/bin/bash
```

- **Rainbow Tables:** Rainbow tables are precomputed tables or reversing cryptographic hash functions, usually for cracking password hashes. As hashes are one-way algorithm they can be restored to its original state rainbow tables are used to brute force hashes. Rainbow tables can be used for malicious purposes too.
- **Vigenère:** The Vigenère cipher is an encryption technique, based on a keyword and the Vigenère square.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| C | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| D | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| F | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E |
| G | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |
| H | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G |
| I | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H |
| J | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I |
| K | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J |
| L | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K |
| M | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L |
| N | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M |
| O | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| P | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| Q | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
| R | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| S | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| T | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| U | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| V | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
| W | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
| X | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
| Y | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |
| Z | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |

Figure 13: Vigenere square.

Given a plaintext and a keyword, that it is repeated until matching the length of the plaintext, just pair the first letter of the plaintext with the first one from the keyword and write down the matching letter from the square and so on with all the letters, i.e.:

```
Plain text: This is an example.
Key Word: MyKey
Ciphertext: Ffs w ge yx ivmkzpc.
```

The ciphertext can be converted back to the plain text without knowing the keyword, given enough long ciphertext is quite straightforward. Firstly is needed to guess the length of the keyword, it can be easily done because words will be repeated and sometimes they will be encrypted using the same

letters, this is called the Kasiski examination. Now the length of the keyword (n) is known, the text can be divided by into n parts. Those parts are Caesar cryptograms that are easy to solve. Furthermore, there are tools like The Black Chamber that allows cracking ciphertexts.

- **SQL Injection:** The SQL injection is a code injection technique used mostly to retrieve information or delete a database. Is one of the most common attack even nowadays as shown in 14 ⁷.

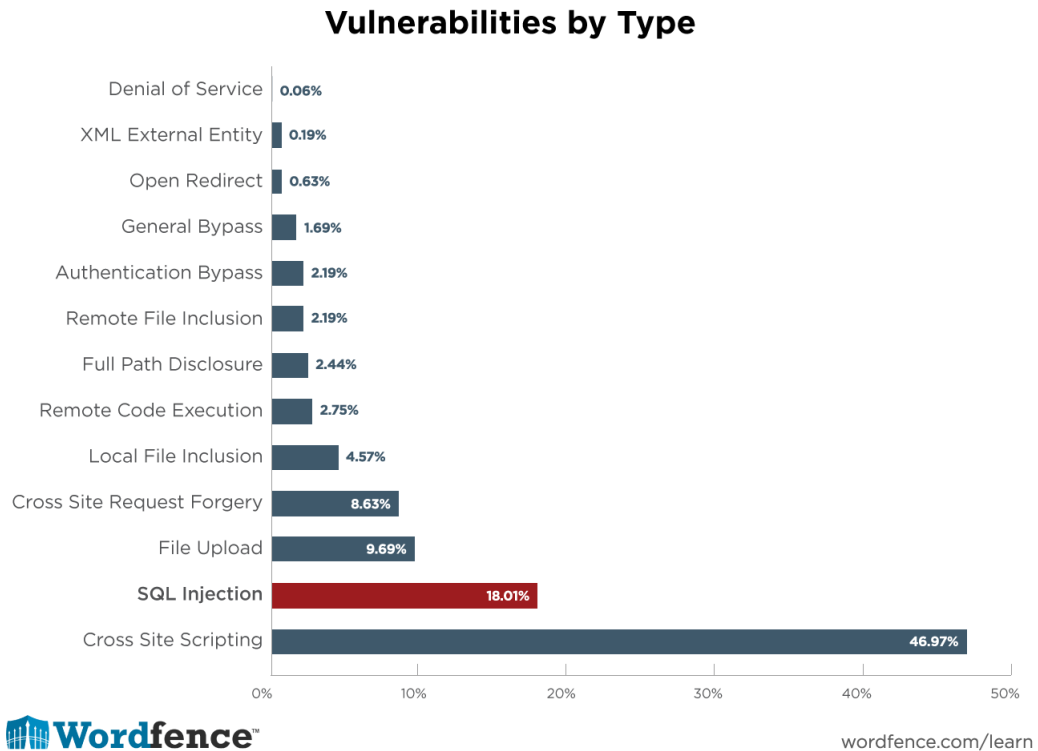


Figure 14: SQL Injection use nowadays.

The code injection occurs when the input data in a web page is not correctly sanitized and it is used as a variable in a SQL sentence. In consequence there is a SQL sentence that has been modified. Using UNION sentences the attacker can even get information from different tables. One of the most common SQL injection is used for accessing systems bypassing the username and password. This is done by cheating the SQL SELECT sentence which is used to verify if a user is in the system, i.e.:

```
#Correctly executed SQL sentence
```

⁷Image taken from Word Fence

```

SELECT * FROM users WHERE username = 'Gorka' AND
password = 'mystrongpassword';

#Code injection. Note that the password input is 'OR '1' = '1

SELECT * FROM users WHERE username = 'Gorka' AND
password = ' 'OR '1' = '1 ';

#SQL interpretation
SELECT * FROM users WHERE username = 'Gorka' AND
password = ' ' OR '1' = '1';
#username = 'Gorka' is true, the username exists
#the password = '' is false, it does not exist for that user
#but '1' = '1' which is true
#so true AND (false OR true) = true AND true = true
#and the access will be granted

```

This methodology can lead to more complex attacks. For further reading check: [SQL injection cheat sheet](#). [1]

- **Cracking Linux passwords:** Password in Linux systems are stored at `/etc/shadow` hashed and encrypted depending on the distribution. Those passwords cannot be read by anyone but the root user. Weak passwords can be broken using an open source tool called John The Ripper. Firstly install John the ripper if is not, depending on the OS.

```

#Ubuntu
$ sudo apt-get install john

```

```

#Arch Linux
pacman -S john

```

Passwords can be cracked using John directly in the `/etc/shadow` file, or using a John command, `unshadow`. `Unshadow` combines `/etc/passwd` file and `/etc/shadow` allowing John to crack passwords too. Note that this example is for Linux systems.

```

#Unshadow password files
sudo unshadow /etc/passwd /etc/shadow > mypasswd.txt

```

```

#Crack the passwords
john --show mypasswd.txt

```

There are also different ways to crack passwords using John, such as dictionary based or incremental cracking. For further reading check [John The Ripper](#)

web page.

7.2. Server

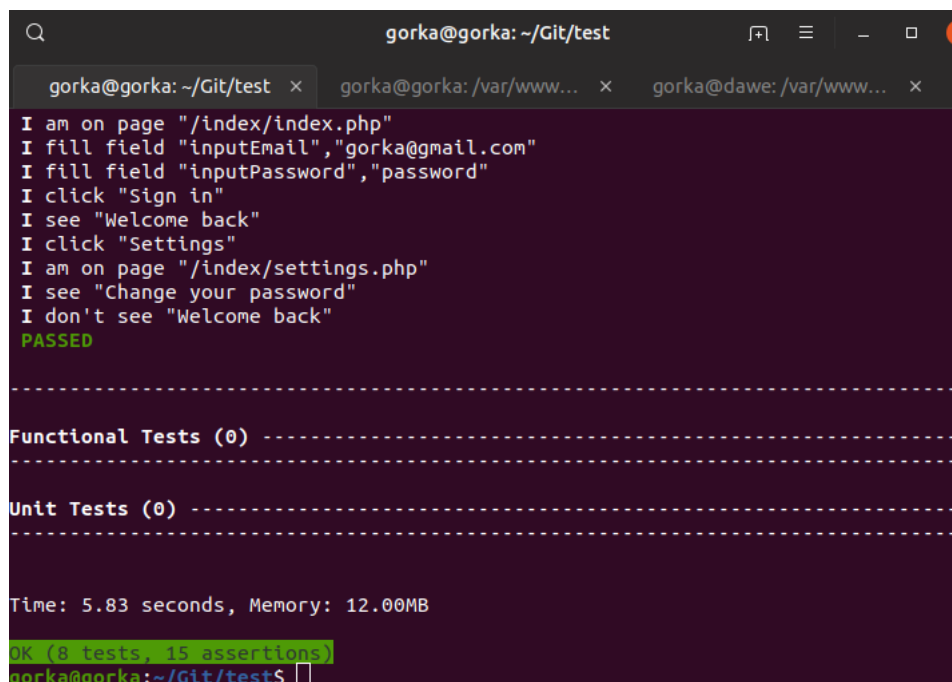
In regards to cloud computing Google Cloud is used, where a compute engine instance was created. This instance is a 1 vCPU, 1,7 GB RAM computer running over Ubuntu 18.04.2 LTS minimal. According to Google Cloud, it was chosen because we already have a piece of previous knowledge about it and because Google offered a learning license in which we could use the server without any cost. In another scope, the power needed to run the applications is more than enough with the cited above specs. Although is true that if more Docker containers will be used, the specs of the computer will have to be improved.

- **Problems:**

We did not encounter any major problem using the server, the configuration was quite straightforward as expected. Moreover, the documentation and the information about Google Cloud is immense and any inconvenience can be easily solved.

8. Testing

For the testing, a framework called Codeception was used. Codeception is a framework that is based in PHP Unit and grants automatic testing for PHP applications. In the appendix at B.5 it is explained how to install and use it. The testing scope reaches every single page of the web application testing the correct functioning of the database. Trying the login with some fixed credentials or testing every challenge. Figure 15 shows the output for a successful testing, and the Figure 16 shows a failed testing.



```
gorka@gorka: ~/Git/test
I am on page "/index/index.php"
I fill field "inputEmail","gorka@gmail.com"
I fill field "inputPassword","password"
I click "Sign in"
I see "Welcome back"
I click "Settings"
I am on page "/index/settings.php"
I see "Change your password"
I don't see "Welcome back"
PASSED

-----
Functional Tests (0) -----
-----
Unit Tests (0) -----
-----

Time: 5.83 seconds, Memory: 12.00MB
OK (8 tests, 15 assertions)
gorka@gorka:~/Git/test$
```

Figure 15: Successfully testing.

```
gorka@gorka: ~/Git/test
gorka@gorka: ~/Git/test x gorka@gorka: /var/www... x gorka@dawe: /var/www... x
Step Don't see "Settings"
Fail Failed asserting that on page /tfg/index/settings.php
--> Settings 50 points Home (current) Gorka Leaderboard Settings Log out Sett
gs Privacy Change your password Password Change Password Change your email Email
Change Email Change your name Name Change Name Made with by Gorka Abad
--> does not contain "Settings".

Scenario Steps:

9. $I->dontSee("Settings") at tests/acceptance/FirstCest.php:79
8. $I->see("Change your password") at tests/acceptance/FirstCest.php:78
7. $I->amOnPage("/index/settings.php") at tests/acceptance/FirstCest.php:77
6. $I->click("Settings") at tests/acceptance/FirstCest.php:76
5. $I->see("Welcome back") at tests/acceptance/FirstCest.php:75
4. $I->click("Sign in") at tests/acceptance/FirstCest.php:74

Artifacts:

Response: /home/gorka/Git/test/tests/_output/FirstCest.settingspageWorks.fail.ht
ml

FAILURES!
Tests: 8, Assertions: 15, Failures: 1.
gorka@gorka:~/Git/test$
```

Figure 16: Failed testing.

Note whenever a test fails, extra information about the test is saved in the output/ directory.

Codeception acceptance testing is easy to work with. Firstly is needed to create a class that will wrap every test case. The functions inside the class are the test cases, there are some worth to know functions such as:

- `$I->amOnPage('path to a resource')` redirects the browser to the given resource.
- `$I->fillField('inputEmail', 'gorka@gmail.com')` fills the input type field with "gorka@gmail.com" identified by "inputEmail".
- `$I->click('Sign in')` clicks in a button or a Link with the text Sign in.
- `$I->see('Hello')` checks if the string "Hello" appears in the web page.
- `$I->dontSee('Bye')` checks if the string "Bye" does not appear in the web page.

9. Future work

As as most of the back-end has been developed using an ad-hoc approach, without drawing on any libraries, this is a clear point of possible enhancement. Also, they ensure the integrity of the tasks developed by those third-party technologies. Those libraries are Symfony or Laravel. They are used globally by thousands of companies, which makes them trustable. Nevertheless, there are other kinds of technologies used nowadays such as using JavaScript for the back end. JavaScript provides lots of libraries that can be used, also there are frameworks like React that could fit in.

In another scope, there is the containerization or the use of containers. As explained, Docker has been chosen for this use. Docker can be used at the same time with Kubernetes, which is a system for container orchestration. It can be used to monitor all the containers with a web page interface, also it can be useful for deploying new containers or updating old ones using rolling updates. This could be a must implementation for future work [6] [8] [11].

Every user can access challenges, they need to have a personal machine for each challenge. Those machines as are generated using Docker, each container needs a port assignment. So for the user 1, the port that has been assigned for the challenge number one is the 9998, for example, ⁸. As the numbers of ports are limited a user could guess another users' machine. Also if there were a huge amount of users, it could lead to a port collision, because the number of ports available is less than the number of users multiplied by the number of challenges. Moreover, the use of ports has to be managed in order to solve ports collision with users and challenges as explained before. A solution has not been found yet.

Also as the project is designed, challenges can be added easily. Challenges may vary, from different difficulty or scope. For example a Cross Site Scripting (XSS) challenge or a challenge in which the 4-way handshake of the WPA2 has to be broken in order to get into the network. Once inside the computer analyze the traffic to retrieve some credentials. Lots of challenges can be created, furthermore creating a guide to solve them could be an interesting feature, it could help to learn.

⁸Ports assignment in this project are from 8000 to 9999

10. Conclusions

Finally, in this last section, the results achieved will be reflected compared to the ones expected.

10.1. Goals

Every single goal has been achieved. An online laboratory for web penetration testing has been made, using the latest containerization technologies. Also, it has been prepared smartly for future updates or upgrades.

Comparing to the requirements defined at 4 to the ones achieved every requirement has been done.

- Challenges have been created using containers so more can be added easily. See section B.3
- Every user has its own profile, where they can see their score and the challenges available. 20
- Leaderboard was made to show the best users. See section B.4.
- Points will be achieved when a user completes a challenge, the points can be seen in the leaderboard or in the navigation bar of each user. B.4
- Log in and Sign up: Log in and Sign up could have been done using a PHP framework as explained at 9 but conversely it has been developed entirely 17.
- Password recovery: A password recovery system has been created from scratch as is explained at 7.
- Modularity: Docker allows creating more containers and adding them easily to the web application as a challenge.

10.2. Time scheduling

Comparing the estimation done at the beginning of the project with the real-time invested has been nearly the same. Differing from the first one for just 25 hours. That time exceed was due to several problems encountered during the development. Also, there have been some problems when changing the environment from local to an online server.

The workload has been more intense at the end of the project than at the beginning. Although the workflow has been constant and quite well distributed.

In the table 3 the time invested for the project development and the the estimation are compared.

Table 3: Time estimation and invested time

| Activities | Estimation (in hours) | Invested (in hours) |
|----------------------------|------------------------------|----------------------------|
| Management | 29 | 28 |
| Meetings | 7 | 7 |
| Code version control | 10 | 8 |
| Define goals and scope | 2 | 3 |
| Tools management | 10 | 10 |
| Learning | 100 | 93 |
| PHP | 25 | 23 |
| Docker | 25 | 30 |
| Boostrap | 25 | 20 |
| Java Script | 25 | 20 |
| Analysis and design | 24 | 32 |
| Data base design | 5 | 5 |
| Web application design | 12 | 20 |
| Choosing tools | 2 | 2 |
| Defining requirements | 5 | 5 |
| Implementation | 255 | 310 |
| Web application | 125 | 150 |
| Docker containers | 100 | 130 |
| Data base | 5 | 5 |
| Server | 25 | 25 |
| Test | 8 | 9 |
| Web application | 5 | 5 |
| Docker containers | 2 | 3 |
| Database | 1 | 1 |
| Documentation | 90 | 90 |
| Technical Report | 75 | 75 |
| Presentation | 15 | 15 |
| Total | 506 | 530 |

References

- [1] Chris Anley. Advanced sql injection in sql server applications. 2002.
- [2] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.
- [3] Cloudways. Deploying php apps with docker. 2017.
- [4] Phil Sturgeon Josh Lockhart et al. *PHP The Right Way*. 2019.
- [5] Katacoda. *Docker courses*.
- [6] Miika Moilanen et al. Deploying an application using docker and kubernetes. 2018.
- [7] OPSXCQ. Shellshock exploit. 2017.
- [8] Gigi Sayfan. *Mastering Kubernetes*. Packt Publishing Ltd, 2017.
- [9] Aly Sivji. Php + mysql using docker compose. 2018.
- [10] James Turnbull. *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2014.
- [11] Deepak Vohra. *Kubernetes microservices with Docker*. Apress, 2016.

A. Installation guide

Here are explained the steps to follow in order to install the web application on a system with Ubuntu 18.04, installation may be similar in other Linux systems.

A.1. Prerequisites

Note that before any installation

```
sudo apt update
```

should be executed, to get every package up to date.

- Ubuntu 18.04: Get Ubuntu 18.04 OS installed or a similar distro.
- PHP 7: Install PHP 7 or a newer version:

```
sudo apt-get install php7.0
```

- Apache: Install Apache2 as a web server:

```
sudo apt-get install apache2
```

- MySQL: Install MySQL latest version

```
sudo apt-get install mysql-server
```

- Docker: Install the containerization environment, in this case Docker.

```
sudo apt install docker
```

- Git: Install Git in order to clone the repository used for this project.

```
sudo apt install git-all
```


A.2. Installing the web application

Clone the repository from Git Hub:

```
git clone https://github.com/GorkaAbad/Penetration-testing-laboratory
```

So the Apache server by default only serves the files stored at `/var/www`. The document root can be changed in the Apache configuration file. Also, files can be moved to the document root directory and Apache will serve them. So moving file to the `/var/www` directory.

```
mv . /var/www/html/
```

Now navigate to localhost if Apache is running locally or to the machine IP and the login screen may appear.

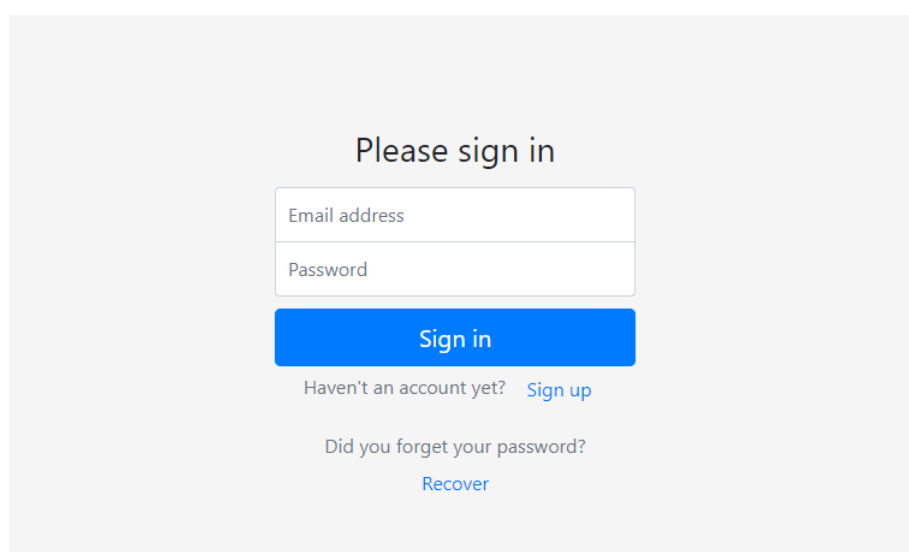


Figure 17: Login page.

Create a database with `db` as its name. Firstly log in in MySQL.

```
mysql -u <your user> -p
```

```
CREATE DATABASE db;
```

Also import the database schema.

```
mysql -u <your user> -p db < db.sql
```

The MySQL username, password, and the database name should be changed in the `database.php` file, according to your preferences.

Additionally, for database management PHP my admin could be installed, but it is not necessary. For the installation follow the instructions below:

```
sudo apt install phpmyadmin
```

Make sure the Docker containers are running, if you have created new ones add them to the database so they can be displayed in the web app. As challenges are linked to the server IP it has to be changed, so in the database container table modify the entry of the `command_container` to your server IP.

Also for the password recovery system, an API Key is needed from Sendgrid. Create an account on it and modify the API Key in the `sengrid.env` file with yours.

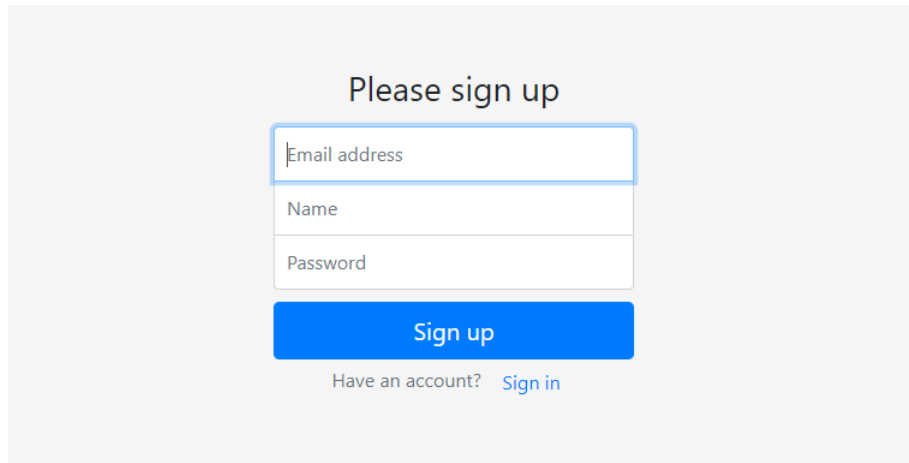
Finally, ensure that ports 80 or 443 and from 8000 to 9999 are open. 80 is for serving web pages over HTTP protocol and 443 for https protocol. 8000 to 9999 will be used for each user in different challenges.

B. User guide

A brief trip over every interface explaining every single functionality of them.

B.1. Login and Sign Up

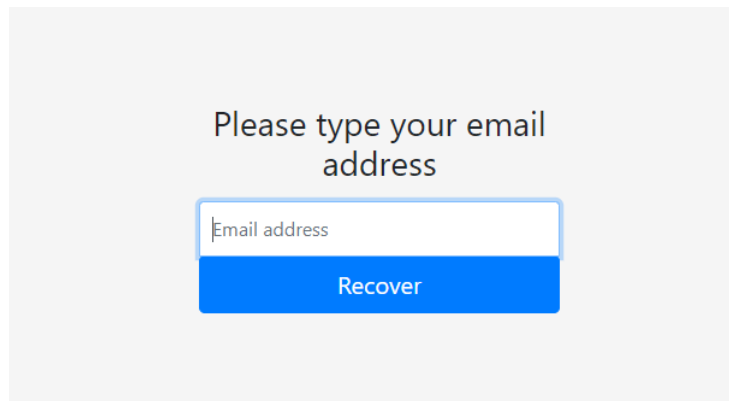
As seen in the 17 is the login page. The signup button redirects the user to a web page where a new user can be signed up.



The image shows a sign-up form titled "Please sign up". It contains three input fields: "Email address", "Name", and "Password". Below the fields is a blue "Sign up" button. At the bottom, there is a link that says "Have an account? Sign in".

Figure 18: Sign up page.

Also, there is a button where the user can recover their password by clicking into the Recover button.



The image shows a password recovery form titled "Please type your email address". It contains one input field labeled "Email address". Below the field is a blue "Recover" button.

Figure 19: Password recovery page.

B.2. Main page

After successfully login in, it is the main page. From there can be executed all the main features of the web application.

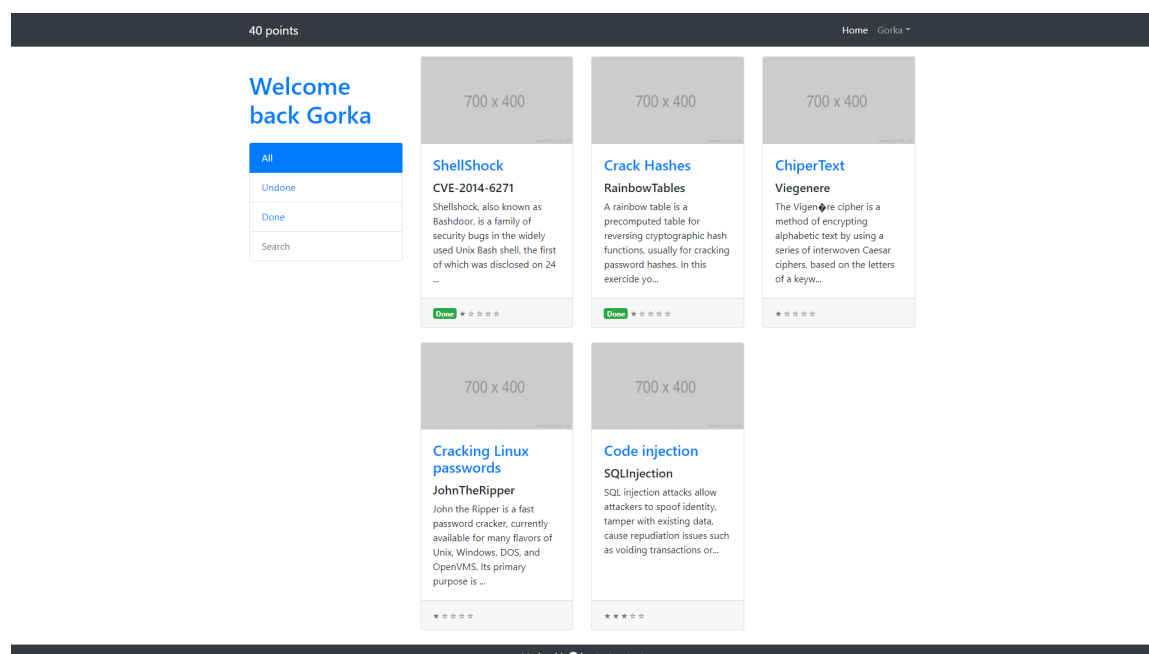


Figure 20: Main page.

At the navigation bar at the top of the page can be found 3 important spots. The navigation bar will be at the top in every single web page.

- Score: At the left, there is the actual score of the current user.
- Home button: At the right, there is a button that redirects the user to the home page.
- Menu: At the very right there is the menu, which if it is clicked appears a teardown menu with 3 different options. Also, it has the name of the current user.

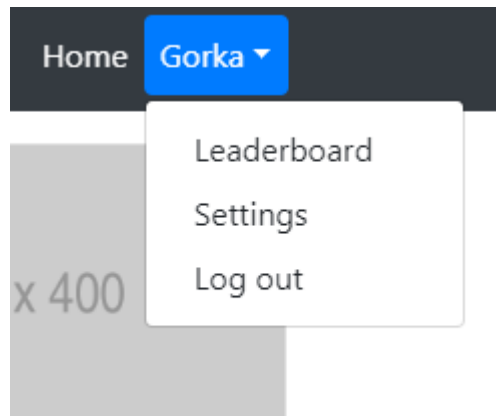


Figure 21: Tear down menu.

- Leaderboard explained at: B.4
- Settings explained at: B.5
- Log out: If clicked the session of the current user gets destroyed and gets redirected to the login page.

At the left part of the page, there are 3 buttons and a search bar.

- The All button show all the challenges available.
- The Undone button show only the challenges that have not been completed by the user.
- The Done button show only the challenges that have been achieved or completed.
- The search bar is used to filter challenges by name.

The middle of the web page are the challenges. Each challenge has an image, a name, an ID, a description a difficulty and a Done flag. Note that the Done flag will only appear once the challenge has been passed.

B.3. Challenge

Every challenge has its description, name, and instructions to follow in order to solve it. There are challenges that require a prepared environment to solve them, in that case, a clickable test will appear at the top right corner, that will automatically prepare the environment for each user.

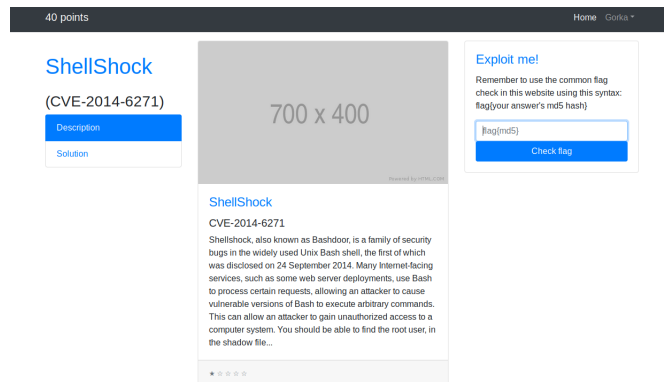


Figure 22: Challenge.

Note that for submitting an answer is need the MD5 hash of the flag and submitting in the correct form: flagMD5 hash. If the answer is correct the score will be increased and the challenge will be marked as done. A challenge can only be solved once if it is already solved the Check flag button will be deactivated.

B.4. Leaderboard

The leaderboard is where the 10 best users appear. Every row contains the number in the ranking, the user name, and the score. Note that the current user in the ranking will be remarked. Also, every user has a profile that can be accessed by clicking in the name.

| # | Name | Points |
|---|-------------------------------------|--------|
| 1 | Gorka | 40 |
| 2 | Unai López Ansolega | 0 |

Figure 23: Leaderboard.

B.5. Settings

The settings section is where the user is able to change their credentials, such as the password, the email or the name.

Settings

Privacy

Change your password

Old Password

New Password

New Password

Change Password

Change your email

Old Email

New Email

New Email

Change Email

Change your name

Old Name

New Name

New Name

Change Name

Figure 24: Settings.

A. Testing guide

A.1. Codeception installation and configuration

For the testing Codeception has been used, the installation is straightforward but it is going to be explained to fit this project configuration.

```
cd /tmp
```

Download de .phar file

```
sudo curl -Ls https://codeception.com/codecept.phar -o /usr/local/bin/codecept
```

Grant execution permissions

```
sudo chmod a+x /usr/local/bin/codecept
```

Move to the directory that is wanted to work in

This creates a configuration file a tests directory
codecept bootstrap

Generate the acceptance test

```
codecept generate:cest acceptance First
```

Make sure that the server is wanted to test is running

Modify the URL of the App in the tests/acceptance.suite.yml file.

```
actor: AcceptanceTester
modules:
enabled:
  - PhpBrowser:
      url: {YOUR APP'S URL}
  - \Helper\Acceptance
```

Modify the content of the tests/acceptance/FirstCest.php file with below

```
<?php
class FirstCest
{
    public function logInpageWorks(AcceptanceTester $I)
    {
        $I->amOnPage('login.php');
        $I->see('Sign');
    }
}
```



```
}?>
```

Execute the test

```
codecept run --steps
```

The code in the `logInpageWorks` function checks if in the login page appears the Sign in button.

All the test done are in the Github repository in the test directory. For further reading about the methods that can be used in the testing please refer to the Cocedption documentation.