ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

Diseño e implementación de una plataforma web de iniciación en la ciberseguridad

Design and implementation of a cybersecurity initiation web platform

Realizado por
Antonio Javier Galán Herrera
Tutorizado por
José Antonio Onieva González
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2023

Fecha defensa:
El Secretario del Tribunal

Agradecimientos

Este ha sido un largo viaje, por lo que me gustaría expresar mi más sincero agradecimiento a todas las personas que estuvieron en él. Su apoyo y colaboración han formado parte de este proyecto, y quiero reconocer su generosidad.

En primer lugar, me gustaría agradecer a mi tutor, José Antonio Onieva González, ya que me resulta imposible pasar por alto lo paciente y considerado que ha sido conmigo, que siempre estuviera disponible para resolver mis dudas y lo útiles y satisfactorias que fueron nuestras reuniones. Ha sido un placer trabajar con él.

También me gustaría dar las gracias a mis compañeros de 42 Málaga, quienes jugaron un gran papel en la fase inicial de este proyecto compartiendo ideas y debates, y durante su desarrollo animándome de forma constante; debo hacer una mención especial a mi compañero Francisco, cuyos aportes fueron de gran ayuda en el desarrollo de este proyecto y cuya experiencia valoro mucho.

Además, quiero destacar a Jesús, compañero de facultad, quien me ofreció guía y recursos cuando lo necesitaba aún cuando él mismo se encontraba terminando su Trabajo de Fin de Máster.

Y por último, pero no menos importante, a todos los que alguna vez contribuyeron de alguna manera a mi progreso en la carrera, ya fuera con consejos, apoyo moral o una mano amiga, les agradezco de corazón.

Gracias a todos por ser parte de este viaje y por haberme ayudado a alcanzar este hito en mi vida académica.

- Antonio Javier Galán Herrera

Resumen:

El objetivo principal de este trabajo de fin de grado es la realización de una plataforma web de entrenamiento e iniciación en el campo de la ciberseguridad con la que futuros estudiantes del sector puedan dar sus primeros pasos.

La plataforma abarcará conceptos relacionados con el ámbito del pentesting y permitirá a los usuarios acceder a entornos virtualizados -laboratorios- donde poner en práctica los conocimientos adquiridos.

Palabras claves: web, ciberseguridad, pentesting, virtualización, contenedores

---

Abstract:

The main objective of this final degree project is the development of a web platform for training and initiation in the field of cybersecurity with which future students in the sector can take their first steps.

The platform will cover concepts related to the field of pentesting and will allow users to access virtualized environments -laboratories- where they can put into practice the knowledge acquired.

Keywords: web, cybersecurity, pentesting, virtualization, containers

Índice de contenidos

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

1.1. Motivación

La Agencia Digital de Andalucía presentó en 2021 un plan de inversión para los años 2022-2025 que se centra en varias áreas estratégicas de desarrollo tecnológico, incluyendo la ciberseguridad. Esta inversión del Gobierno andaluz no solo impulsará la creación de un Centro de Ciberseguridad que coordinará la Estrategia Andaluza de Ciberseguridad, sino que además, pretende conseguir que revierta en el desarrollo económico de la región, que ya ha empezado a aceptar empresas como Google, que ha instalado su Centro de Excelencia de Ciberseguridad en Málaga.

Bajo el paraguas de este desarrollo tecnológico, el aumento de la demanda de profesionales en este campo ha despertado un gran interés en los estudiantes por aprender sobre la ciberseguridad.

Sin embargo, muchos de ellos se enfrentan a la dificultad de encontrar recursos educativos completos y accesibles debido a las restricciones que imponen algunas plataformas de entrenamiento, donde los recursos que ofrecen tienen barreras que dificultan el acceso a la formación, incluyendo la necesidad de suscripciones para acceder al contenido completo.

Esto puede desmotivar a los estudiantes y reducir su capacidad para aprender y desarrollar habilidades en el campo de la ciberseguridad.

1.2. Objetivos

Se busca desarrollar una plataforma web que sea ligera, gratuita, *open-source* y fácilmente extensible, con el objetivo principal de simplificar la introducción de futuros estudiantes en el sector de la ciberseguridad, con un enfoque específico en el pentesting.

La plataforma proporcionará información documentada y sintetizada junto con entornos de prueba virtualizados (laboratorios) para facilitar el aprendizaje mediante la experimentación.

A diferencia de otras plataformas como Hack The Box y similares, el enfoque de esta plataforma no estará en la resolución de retos, como serían las pruebas de tipo CTF

más habituales en el sector, sino en guiar e instruir a los usuarios de la plataforma sobre distintos conceptos y técnicas de pentesting, proporcionando entornos virtuales diseñados específicamente para experimentar y poner en práctica conceptos concretos.

Esta herramienta brindará a los usuarios una experiencia educativa gratuita y accesible en ciberseguridad, permitiéndoles aprender y desarrollar habilidades prácticas de manera efectiva y en línea con las necesidades del mercado.

1.3. Metodología

Se ha empleado un desarrollo incremental para la elaboración del proyecto: se partió de una base a la que se añadieron sucesivas mejoras sin que estas perjudicaran al resto del proyecto desarrollado hasta ese momento.

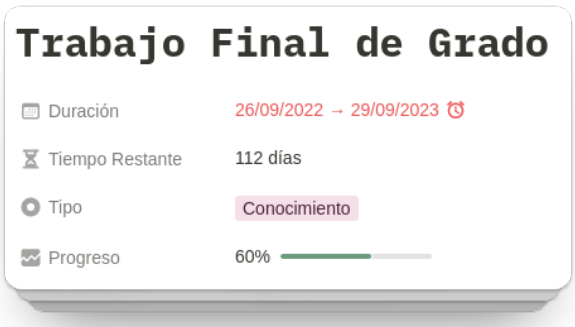
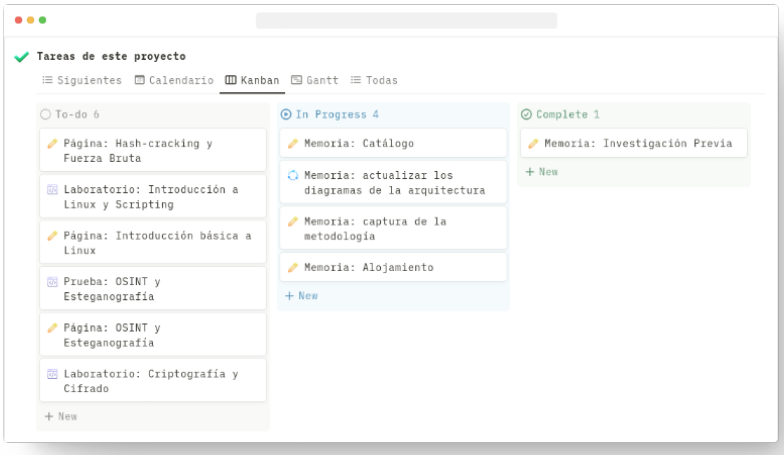


Figura 1.1: Resumen del estado del proyecto en Notion al momento de la captura

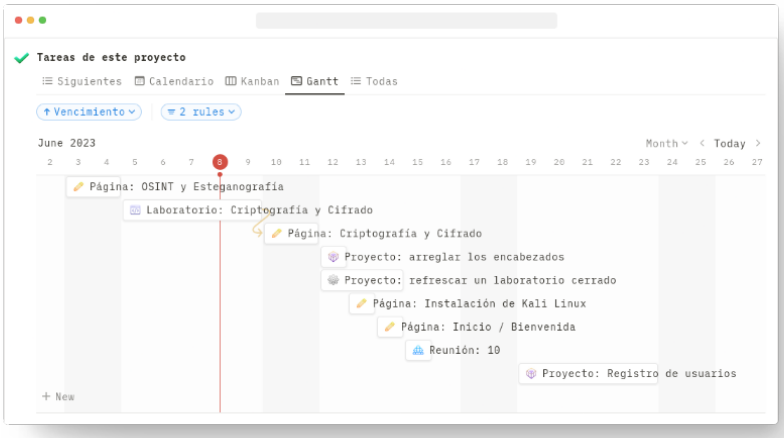
Se ha utilizado la herramienta Notion para llevar a cabo este proyecto, una aplicación que permite la creación de notas -llamadas páginas- cuyo contenido puede ser variado: texto, imágenes, tablas, listas, archivos, enlaces, etc. y que además, permite la creación de bases de datos que pueden ser utilizadas para definir tablones Kanban, diagramas de Gantt, y otros tipos de vistas dinámicas.

Notion también cuenta con otra funcionalidad recientemente añadida: la integración con GitHub [1]. Esto permite añadir a cualquier página de Notion una base de datos generada a partir del estado de un repositorio remoto -*Issues*, *Projects* y *Pull Requests*-, permitiendo además la manipulación y la actualización del contenido en ambas direcciones, todo sincronizado en tiempo real.

El proceso que se ha seguido para el desarrollo del proyecto ha sido definir tareas en Notion y descomponerlas en tareas más simples, atómicas y tratables, clasificadas en función de su propósito: componer la memoria, desarrollar la plataforma web o crear entornos de pruebas virtualizados (sección 1.4). Por otra parte, se han asignado dependencias entre las tareas, por lo que en la mayoría de ocasiones resultaba intuitivo y visual establecer el orden que debían seguir dichas tareas.



(a) Vista de Kanban



(b) Vista de Gantt

Figura 1.2: Gestión de tareas del proyecto en una base de datos de Notion

El desarrollo se ha llevado a cabo utilizando, principalmente, las vistas de los sistemas: Kanban, con las tareas clasificadas en función de su estado; y Gantt, con las tareas clasificadas y ordenadas en función de sus dependencias. No obstante, también se han utilizado las vistas de Lista para visualizar las tareas de una forma más compacta y ordenada, y de Calendario para visualizar las tareas en función de su fecha de finalización.

1.4. Estructura global del proyecto

El proyecto se divide en tres pilares principales: la creación de una plataforma web, la investigación -y selección- de conceptos de pentesting y la virtualización de los entornos de prueba:

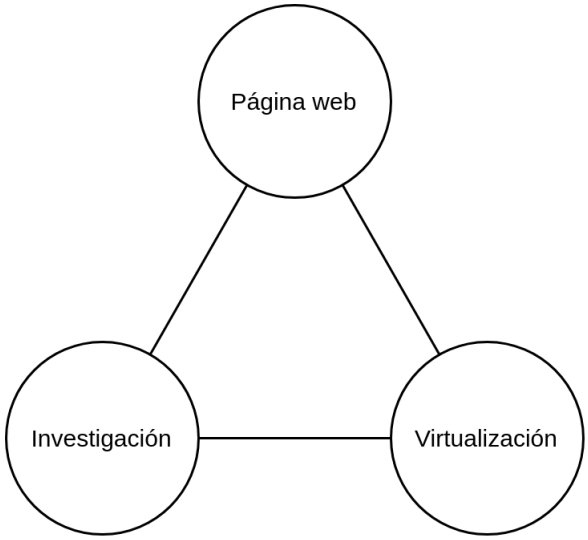


Figura 1.3: Pilares fundamentales del proyecto

1.4.1. Página web

La primera toma de contacto del usuario con la plataforma, la página web será la interfaz de usuario para acceder a los laboratorios y contenidos relacionados con la ciberseguridad.

Respecto a su construcción, se han considerado dos opciones:

- **Programación desde cero**: se consideró el uso del framework Astro, que permite crear sitios web rápidamente, además de proporcionar un amplio conjunto de herramientas para su personalización.
- **Utilizar un CMS (Sistema de Gestión de Contenido)**: se consideró el uso de WordPress, uno de los CMS más utilizados en el mundo y que cuenta con una amplia variedad de plantillas disponibles, lo que facilitaría el diseño de la interfaz.

Finalmente, se decidió usar WordPress por su sencillez y su biblioteca de plugins, que permitiría integrar diferentes funcionalidades extra en lugar de programarlas manualmente como podría haber ocurrido con el uso de un framework.

Alojamiento

Respecto al alojamiento, se han considerado varias opciones, tales como GitHub Pages [2], Vercel [3], OVHcloud [4] o Heroku [5].

Sin embargo, al tratarse de una prueba de concepto, se ha decidido elaborar la infraestructura de forma local, lo que permitirá un mayor control y flexibilidad en el desarrollo, dando lugar a mejoras en futuras líneas de trabajo (sección 5.3).

En cuanto a la base de datos, se ha decidido utilizar MySQL por ser una opción muy utilizada, fácil de configurar y ser nativamente compatible con WordPress. Se usará la tabla `wp_usermeta` para almacenar la información de los laboratorios activos.

1.4.2. Investigación

Consistirá en recopilar y seleccionar diferentes conceptos y técnicas relacionados con el pentesting. Este apartado se verá mejor reflejado en la sección 4.

Si bien la investigación se plantea para una fase inicial del proyecto, igualmente se espera que se lleve a cabo de forma continua en paralelo con el desarrollo de la plataforma, ya que los resultados de la investigación se llevarán a cabo para la creación de más laboratorios.

1.4.3. Virtualización

La virtualización es una parte integral del proyecto, ya que permite la creación de varios laboratorios de prueba donde los usuarios aplicarán los conocimientos adquiridos en la plataforma. Concretamente, se consideraron contenedores de Docker y máquinas virtuales para crear estos laboratorios.

Elegir Docker como plataforma de virtualización es una buena opción, porque ofrece una gestión de recursos eficiente y una generación de laboratorios sencilla a través de imágenes y contenedores; por otro lado, Vagrant Cloud [6] es una buena opción para crear máquinas virtuales, ya que ofrece una amplia gama de imágenes preconfiguradas.

Por otro lado, también se consideraron plataformas que brindan servicios de almacenamiento y distribución de imágenes, como Docker Hub [7] para imágenes de contenedores de Docker y Vagrant Cloud para máquinas virtuales preconfiguradas. Estas dos plataformas brindan enlaces de descargas con las que un usuario puede acceder a recursos previamente construidos por su autor.

1.5. Tecnologías empleadas

Esta sección recoge de forma directa las tecnologías y herramientas que se usaron para llevar a cabo el proyecto. Algunas de ellas fueron una elección clara, mientras que otras pasaron por un proceso de análisis, recogido en la sección 2.3.

1.5.1. GitHub

GitHub es una plataforma de desarrollo colaborativo de software que ofrece alojamiento de código fuente, seguimiento de problemas y herramientas de colaboración para proyectos de programación.

Esta plataforma es muy popular en la comunidad de desarrolladores, ya que les permite compartir su trabajo con otros usuarios, pudiendo ser clonado, modificado y actualizado por otros desarrolladores, permitiendo la colaboración y el intercambio de conocimientos, así como mantenerlo privado para uso personal o empresarial.

Además del alojamiento de código fuente, GitHub también ofrece herramientas de seguimiento de problemas, donde los desarrolladores pueden reportar y resolver errores, solicitar nuevas características y discutir mejoras en el software. Esto facilita la comunicación y la colaboración entre los usuarios, además de mejorar la calidad del software desarrollado.

GitHub se ha empleado para llevar un control de versiones sobre la construcción del sitio web, el desarrollo de los laboratorios de la plataforma, y la elaboración de este mismo documento (a modo de copias de seguridad).

1.5.2. WordPress

WordPress [8] es una plataforma de código abierto para la creación y gestión de sitios web que fue publicada en 2003 y desde entonces, se ha convertido en una de las plataformas más populares y utilizadas en todo el mundo, siendo especialmente popular para la creación de blogs, aunque también es utilizada para sitios web corporativos, tiendas en línea, portafolios, entre otros.

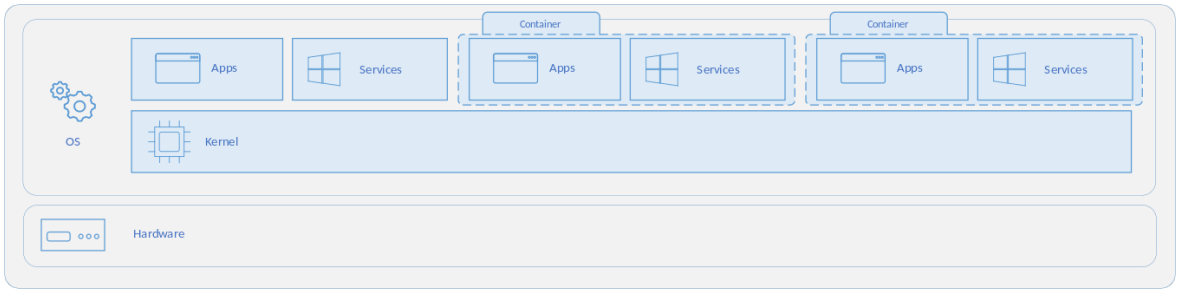
WordPress es utilizado por muchos grandes sitios web y empresas, como el blog de noticias *TIME* [9] y el sitio web de la *BBC America* [10]. También es utilizado por miles de bloggers y pequeñas empresas en todo el mundo debido a su facilidad de uso y capacidad de personalización.

Una de las principales ventajas de WordPress es que es muy fácil de usar y no se requiere experiencia técnica previa para comenzar a utilizarlo. Otra ventaja de WordPress es su personalización gracias a la gran cantidad de temas y complementos (plugins) disponibles. Estos complementos permiten añadir nuevas funcionalidades a un sitio web, como formularios de contacto, galerías de imágenes, integración con redes sociales, etc. evitando la necesidad de programar dichas funcionalidades de forma manual.

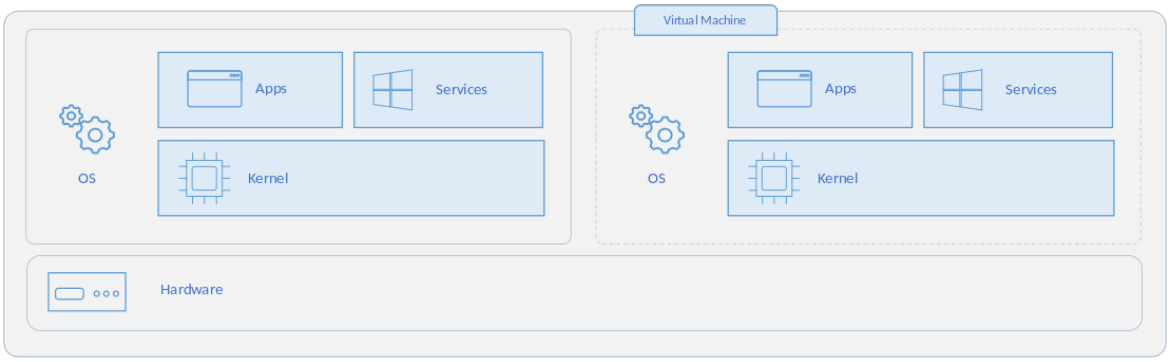
WordPress se ha usado para crear la página web de la plataforma; además, se ha empleado un tema hijo a partir del tema Divi [11], uno de los temas más populares de WordPress y que además, incluye su propio editor de páginas (Divi Builder).

1.5.3. Docker

Docker es una plataforma de software que permite crear, implementar y administrar aplicaciones en contenedores, siendo un contenedor un entorno aislado y seguro que recoge una aplicación y todas sus dependencias, lo que permite que se ejecute sin problemas en cualquier entorno informático, independientemente de las diferencias entre los sistemas operativos o las configuraciones de hardware.



(a) Arquitectura de los contenedores



(b) Arquitectura de las máquinas virtuales

Figura 1.4: Diferencias entre contenedores y las máquinas virtuales

Docker funciona con imágenes, que son plantillas o modelos que contienen todos los componentes necesarios para ejecutar una aplicación en un contenedor, incluidos el código fuente, las librerías, los archivos de configuración y las dependencias del sistema. Estas imágenes pueden ser descargadas desde un registro centralizado llamado Docker Hub o construirse localmente por los desarrolladores.

Una vez obtenida una imagen, se puede crear un contenedor a partir de ella, lo que implica crear una instancia de una aplicación y, por tanto, el entorno en el que se ejecuta. Los contenedores de Docker se pueden transferir fácilmente entre diferentes sistemas y entornos, lo que facilita la implementación y el escalado de aplicaciones.

Además, proporciona herramientas de administración de contenedores, como la capacidad de iniciar, detener, reiniciar y eliminar contenedores, así como la capacidad de monitorear su uso y rendimiento.

Docker se ha empleado para crear los laboratorios de la plataforma; concretamente, cada laboratorio parte de un fichero Dockerfile cuya imagen es generada y almacenada en el sistema, y cuyos contenedores se crean y se destruyen bajo petición del usuario de la plataforma.

1.5.4. LAMP

Este es un acrónimo que se utiliza comúnmente en el mundo de los servidores para describir un conjunto de tecnologías de código abierto que se utilizan para construir y ejecutar aplicaciones web dinámicas.

Componente	Tipo	Descripción
Linux	Sistema Operativo	Utilizado ampliamente en servidores web debido a su fiabilidad, seguridad y flexibilidad.
Apache	Servidor Web	Alojamiento de sitios web y aplicaciones web.
MySQL	RDBMS	Almacenamiento y recuperación de información para aplicaciones web.
PHP	Lenguaje de Programación	Creación de sitios web y aplicaciones web dinámicas. Procesa la lógica de la aplicación en el servidor.

Tabla 1.1: Componentes habituales de la pila de software LAMP

Esta pila ha sido ampliamente adoptada en la comunidad de desarrollo web y ha sido utilizada para crear una amplia variedad de aplicaciones, desde sitios web simples hasta aplicaciones web complejas y de alto tráfico.

Además de los componentes principales mencionados anteriormente, la pila LAMP se puede personalizar y ampliar utilizando una variedad de tecnologías y herramientas adicionales, como frameworks de desarrollo, sistemas de caché, servidores de aplicaciones y más, según las necesidades específicas de cada proyecto.

LAMP se ha empleado para crear la infraestructura de la plataforma web, y esta a su vez se ha desarrollado en un servidor local.

CAPÍTULO 2

Investigación previa

2.1. Estado del arte

Actualmente existen diversas plataformas web de entrenamiento en ciberseguridad que ofrecen laboratorios y desafíos para mejorar las habilidades de seguridad ofensiva y defensiva de los usuarios.

Dichas plataformas han experimentado un crecimiento en popularidad en los últimos años debido a la creciente demanda de profesionales en el campo de la seguridad informática, así como la necesidad de mejorar las habilidades tanto de los estudiantes como de los profesionales en el campo.

Estas plataformas varían en su enfoque y contenido, ofreciendo desde laboratorios que simulan entornos de la vida real hasta desafíos de explotación de vulnerabilidades, y otros formatos de naturaleza más lúdica y competitiva como los retos de capturar la bandera, también conocidos como CTFs (*Capture The Flag*).

Algunas de estas plataformas han sido utilizadas en la educación, ya que permiten a los usuarios practicar y aplicar conceptos teóricos de seguridad informática en un entorno práctico, y también pueden ser útiles para empresas y organizaciones, ya sea para evaluar las habilidades de seguridad de los empleados o para mejorar la seguridad de sus sistemas.

Respecto al futuro, se espera que continúen creciendo en popularidad y que aumente la variedad de laboratorios y desafíos que ofrecen, siendo muy probable que se produzca una mayor integración con otras tecnologías de inteligencia artificial, quizás dando lugar a una mejora en la calidad de esos desafíos y laboratorios mencionados, para que proporcionen una experiencia de usuario más personalizada.

A continuación se presenta una breve descripción de algunas de las plataformas más populares actualmente, donde se muestran sus similitudes y sus diferencias:

2.1.1. Hack The Box

Plataforma web de entrenamiento que ofrece más de 200 desafíos y laboratorios de amplia variedad, diseñados para mejorar las habilidades de seguridad ofensiva y defensiva de los usuarios. Hack The Box se ha vuelto la plataforma más popular en la comunidad de seguridad informática debido a su enfoque en la calidad de los desafíos y laboratorios y su comunidad activa de usuarios.

Sus retos abarcan desde desafíos básicos de hacking hasta laboratorios avanzados, y también ofrece elementos llamados *boxes* diseñados para simular entornos de la vida real, como redes empresariales, que contienen múltiples vulnerabilidades que los usuarios pueden explotar para ganar acceso a sistemas y obtener información confidencial.

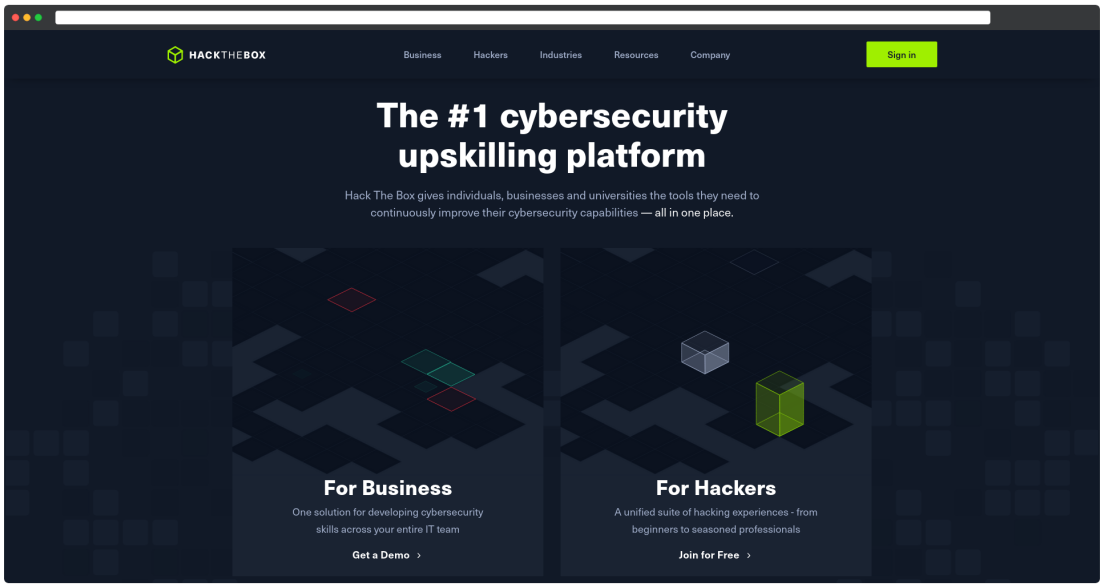


Figura 2.1: Web de Hack The Box [12]

Una de las características más interesantes de esta plataforma es su enfoque en el hacking ético, ya que se fomenta una cultura de hacking responsable y legal y requiere que los usuarios acepten un código de conducta antes de unirse. Los usuarios también son alentados a informar sobre cualquier vulnerabilidad que encuentren en la plataforma.

Hack the Box ofrece diferentes planes de suscripción para los usuarios interesados en acceder a su contenido:

- **Plan gratuito**: acceso limitado a una selección de desafíos y laboratorios, pero no incluye acceso a los *boxes*.
- **Plan VIP**: acceso completo a la plataforma (todos los laboratorios, desafíos y *boxes* disponibles).
- **Planes empresariales**: personalizados para empresas y organizaciones que deseen utilizar la plataforma para la formación y evaluación de sus empleados en seguridad informática.

HTB Academy

Esta es una iniciativa de Hack The Box que también ofrece formación, pero al contrario que Hack The Box, centrada en la práctica y el desafío en tiempo real, HTB Academy se enfoca en la enseñanza práctica de habilidades a través de cursos y laboratorios guiados.

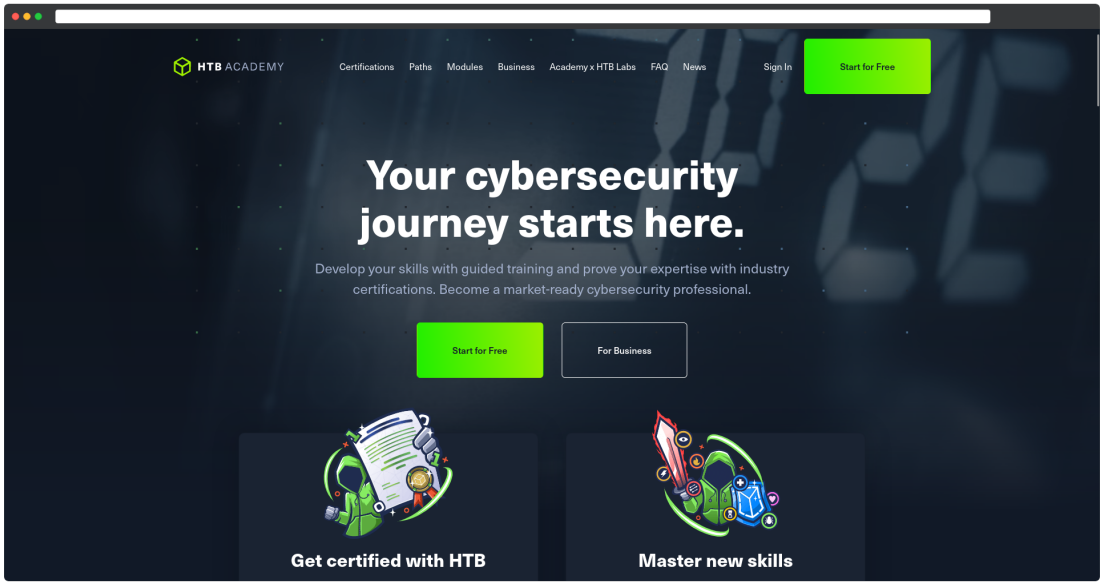


Figura 2.2: Web de HTB Academy [13]

Los cursos están diseñados para ser prácticos, con una orientación en la experimentación activa que permite a los usuarios adquirir habilidades en la práctica, y no solo a través de la teoría.

Cubren una amplia gama de temas, desde los fundamentos de la seguridad informática hasta temas avanzados como el análisis de malware, el hacking web y la ingeniería inversa, ya que están diseñados para ser accesibles desde principiantes hasta expertos en el sector.

Esta plataforma cuenta con los mismos tipos de planes de suscripción que Hack The Box: gratuito, VIP y empresarial; aunque es importante destacar que tanto Hack The Box como HTB Academy son servicios diferentes, por lo que los planes de ambas plataformas son completamente independientes entre sí.

2.1.2. TryHackMe

Plataforma de aprendizaje de ciberseguridad basada en la experimentación activa, que proporciona una variedad de entornos de laboratorios virtuales guiados y desafíos prácticos para ayudar a los usuarios a aprender y mejorar sus habilidades en seguridad informática. Su contenido se encuentra organizado en diferentes rutas de aprendizaje que permiten a los usuarios desarrollar su conocimiento de forma estructurada.

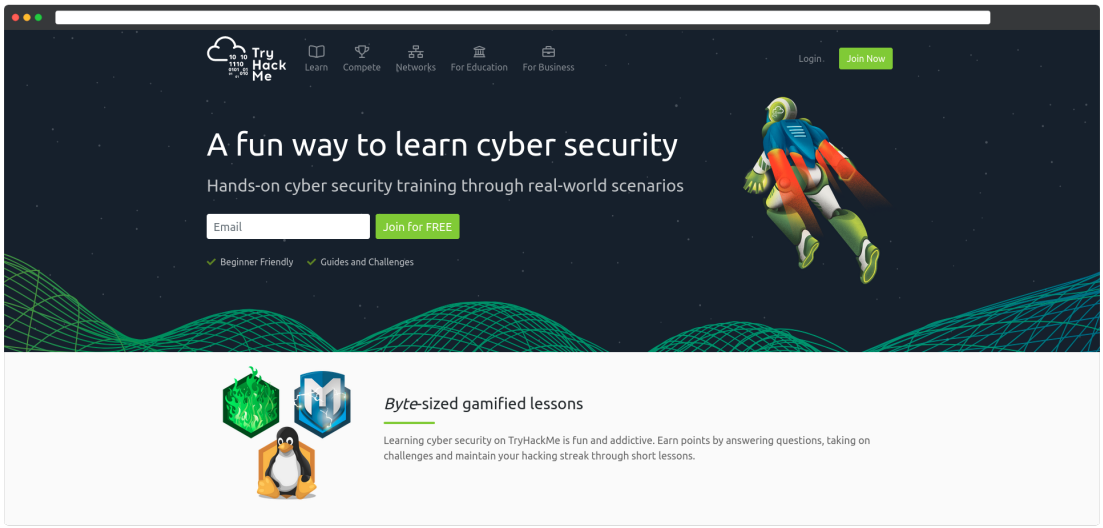


Figura 2.3: Web de TryHackMe [14]

La plataforma también cuenta con una comunidad activa y una función de *gamificación* que proporciona una experiencia de aprendizaje más interactiva y entretenida, permitiendo que los usuarios puedan competir entre ellos, ganando puntos y recompensas por completar desafíos y resolver problemas de seguridad.

TryHackMe ofrece diferentes planes de suscripción para los usuarios interesados:

- **Plan gratuito**: acceso limitado a una selección de desafíos y laboratorios.
- **Plan premium**: acceso completo a la plataforma (todos los laboratorios y desafíos).
- **Planes empresariales**: personalizados para empresas y organizaciones que deseen utilizar la plataforma para la formación y evaluación de sus empleados en seguridad informática.

2.1.3. VulnHub

Plataforma de laboratorios que proporciona una gran cantidad de máquinas virtuales vulnerables que los usuarios pueden descargar y configurar en sus propios entornos de laboratorio para luego explotar sus vulnerabilidades. Cada máquina virtual cuenta con descripción detallada de su objetivo y una guía paso a paso para ayudar a los usuarios en su proceso de aprendizaje.

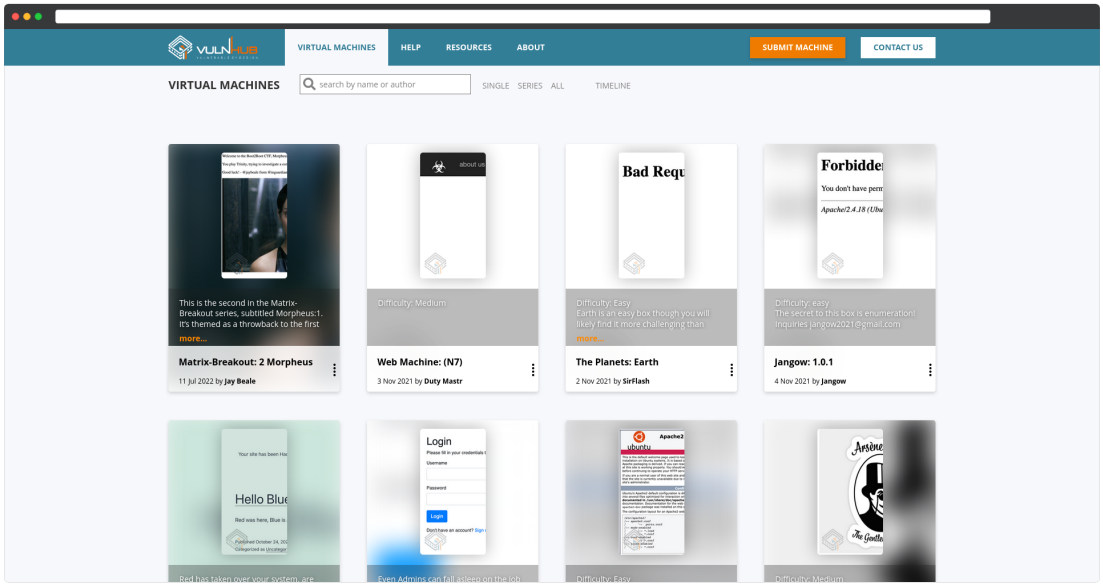


Figura 2.4: Web de VulnHub [15]

Los desafíos de VulnHub son diseñados por la comunidad a través de la opción que tienen los usuarios de poder crear y compartir sus propios desafíos y máquinas virtuales.

Precisamente por eso, al contrario de lo que sucede con otras plataformas mencionadas anteriormente, VulnHub es completamente gratuita y se basa en el principio de ofrecer contenido por y para los usuarios.

2.1.4. OverTheWire

Plataforma de laboratorios diseñada de manera progresiva, donde los usuarios pueden avanzar en su aprendizaje de forma gradual, comenzando con los niveles más fáciles y avanzando hacia los más complejos. Cada nivel de desafío presenta un objetivo diferente, haciendo que los usuarios deban usar su ingenio y habilidades en seguridad informática para resolver los desafíos y avanzar al siguiente nivel.

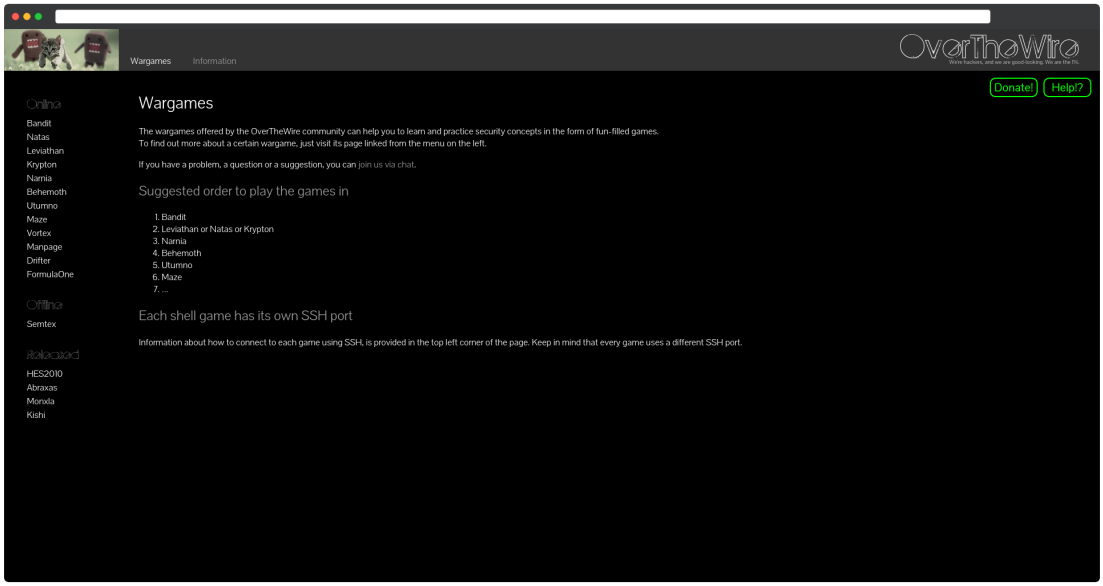


Figura 2.5: Web de OverTheWire [16]

Uno de los aspectos únicos de OverTheWire es que los desafíos están diseñados para simular situaciones del mundo real, lo que permite a los usuarios adquirir habilidades prácticas y relevantes para el mundo laboral de la ciberseguridad, pudiendo aplicar lo aprendido a situaciones reales y utilizar sus habilidades para asegurar sistemas y aplicaciones.

Al contrario de lo que sucede con otras plataformas, entre ellas Hack The Box y TryHackMe mencionadas anteriormente, esta plataforma es completamente gratuita y todo su contenido está construido por y para los usuarios.

2.2. Proceso creativo

Antes de comenzar a desarrollar el proyecto, resulta importante saber de forma más específica *qué* se quiere llevar a cabo, y una vez decidio el objetivo, analizar las posibles opciones para conseguirlo.

Ya se realizó una investigación previa sobre las diferentes plataformas de aprendizaje de ciberseguridad existentes -descritas en la sección anterior (2.1)- con el objetivo de analizar sus características y funcionalidades.

La mayoría -y más conocidas- de estas plataformas son CTFs, y aunque son muy útiles para poner en práctica los conocimientos adquiridos, no suelen ofrecer un entorno de aprendizaje guiado, sino que el usuario debe resolver los desafíos por su cuenta, sin ningún tipo de documentación que le ayude a resolverlos, más allá de las pistas que ofrezca la propia plataforma o las soluciones que hayan publicado otros usuarios en Internet.

Sin embargo, esto no siempre es así, ya que como se vió en la sección anterior, TryHackMe ofrece tanto retos instructivos como retos puramente prácticos y competitivos; mientras que Hack The Box, la plataforma más reconocida, ha sido capaz de lanzar su propia plataforma educativa con retos guiados: HTB Academy.

Teniendo eso en cuenta, se plantearon las siguientes ideas para afrontar el proyecto.

2.2.1. Plataforma de CTFs

La primera idea planteada fue, precisamente, replicar una plataforma de CTFs (Capture The Flag). Esta versión del proyecto se centraría más en enseñar distintos conceptos de la ciberseguridad al usuario, acompañados de una serie de retos que pondrían a prueba los conocimientos adquiridos, parecido a las plataformas TryHackMe y HTB Academy mencionadas en el apartado anterior.

Esta idea fue descartada porque ya existen muchas plataformas de CTFs y se pretendía hacer algo distinto con este proyecto. Se requería un entorno de aprendizaje guiado y como se mencionó anteriormente, los CTFs no suelen ofrecerlo.

2.2.2. Plataforma como portal de descargas

Tomando como ejemplo los retos descargables de la Academia Hacker de Incibe [17], se planteó la posibilidad de crear máquinas virtuales y alojarlas en el servidor a modo de elementos descargables.

Es decir, la plataforma contaría con un sitio web que ofrecería información documentada sobre distintos conceptos de ciberseguridad, pero la virtualización se llevaría a cabo por el propio usuario en su equipo, experimentando localmente.

Esta idea también fue descartada porque plantea un gran problema de fricción entre el usuario y el objetivo de la plataforma: no existiría la posibilidad de ofrecer un entorno de trabajo en el que el usuario pudiera interactuar con los laboratorios de forma remota y con herramientas aisladas, sino que este debería descargar y configurar un hypervisor, o instalar Docker para acceder a los entornos de prueba en su equipo.

Debido a eso, se perdería esa esencia de minimalismo y sencillez que debería ofrecer la plataforma, y además, plantea otros problemas relacionados con el espacio de almacenamiento (tanto en el servidor como en el equipo del usuario).

También puede ser útil incluir en la plataforma de una guía introductoria a la configuración de una máquina virtual para tests de intrusión (pentesing) con Kali Linux, ya que si bien no es objeto de este proyecto que los usuarios aprendan sobre virtualización, no se puede ignorar el hecho de que es necesario el uso de máquinas virtuales para el sector de la ciberseguridad.

2.2.3. Plataforma de laboratorios

Por último, se planteó crear una plataforma de laboratorios con la que el usuario pudiera acceder a distintos entornos de forma remota, sin necesidad de descargar ni configurar nada en su equipo. La plataforma estaría compuesta de un sitio web donde estaría recogida la documentación, mientras que el servidor alojaría los entornos virtualizados asociados a dicha documentación.

La principal diferencia y rasgo a destacar de este planteamiento es que los laboratorios actuarían como *sandboxes*; es decir, ofrecería al usuario un entorno con herramientas preinstaladas en el caso de querer hacer sus propias pruebas sobre un concepto en específico.

Esta idea fue la que más se acercaba a los objetivos del proyecto, y por tanto fue la que se escogió para su desarrollo: solventaba las fricciones de las ideas anteriores, y además ofrecía un entorno de aprendizaje guiado y práctico, que era el objetivo principal del proyecto.

2.2.4. Docker para los laboratorios

Docker, descrito en la sección 1.5.3, permitiría la creación de contenedores para los distintos laboratorios, siendo una solución mucho más ligera que el uso de máquinas virtuales.

Esta idea surgió a partir del artículo *Cyber Security Testbeds and Malware Testing* [18] de la Universidad de Trento, donde un grupo de investigación realizaban un estudio sobre los efectos de un *exploit* de una aplicación sobre varias versiones de la misma y usada en distintos entornos de desarrollo.

> Given a software environment E, an exploit X that successfully subverts an application A, that is running on E:
> - Will X be successful on an application A running on another environment E'?
> - Will X be successful on another version of A, A', running on E?
> - Will X be successful on another version of A, A', running on E'?

Para ello, el grupo de investigación desarrolló un conjunto de pruebas a partir de contenedores, usando combinaciones de un conenedor principal junto a contenedores secundarios.

> Instead of creating separate virtual machines for every application/configuration we use the Linux Containers technology that provides virtualization capabilities on the operation system level.
>
> ...
>
> We use Docker to implement two types of containers: (1) software-specific that contain operating system, webserver and database engine; (2) application-specific that is build on top of a desired software-specific container and also encapsulates the application files. The figure on the right shows an example Wordpress3.2 application container that has been built on top of the "ubuntu-apache-mysql" software container.

Y la figura que representa dicho planteamiento es la siguiente:

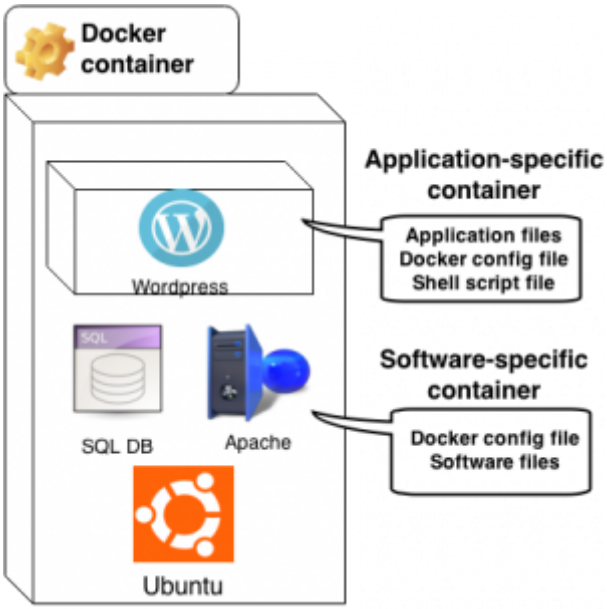


Figura 2.6: Ejemplo de contenedor extraído del artículo [18]

Solución del artículo

La estructura descrita en el artículo se forma a través del uso de la tecnología de contenedores de Linux, alojando cada aplicación y configuración de software en un contenedor.

Una posible réplica de ese mecanismo podría realizarse de la siguiente forma:

1. Crear una imagen específica del software a partir de un archivo Dockerfile que describe la configuración y los componentes que deben incluirse en los contenedores resultantes de dicha imagen.
2. Crear una imagen específica de la aplicación que se construye sobre la imagen del software previamente creada; es decir, se crea una nueva imagen que constituye una capa de configuración adicional sobre la configuración de la propia imagen del software, que actua como base. Esta nueva imagen encapsula los archivos de la aplicación y se crea también a partir de un archivo Dockerfile.

El resultado final es un contenedor específico de la aplicación que reutiliza el sistema operativo y los componentes incluidos en la imagen del software, pero que añade los archivos específicos para el funcionamiento de la aplicación.

A continuación se muestran dos ejemplos de pseudo-código de ficheros Dockerfile que describen la creación de imágenes específicas del software y de la aplicación, respectivamente.

Imagen padre Este Dockerfile establece una configuración básica del sistema operativo, y se instala el software necesario para el funcionamiento de la aplicación. Su propósito es servir como base para la creación de la imagen hija, que contendrá los archivos de la aplicación y se construirá sobre la imagen padre:

```
# Se usa una imagen base con ubuntu
FROM ubuntu:20.04

# Se instala el servidor web Apache
RUN apt-get update && apt-get install apache2 -y

# Se instala el motor de la base de datos MySQL
RUN apt-get install mysql-server -y

# Se especifica la ruta de ejecucion del servidor web
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Listing 2.1: Dockerfile de una imagen principal (padre)

Imagen hija Este Dockerfile hace referencia a la imagen padre mediante la línea `FROM software_container`, estableciendo la dependencia. Además, se copian los archivos de la aplicación en la imagen y se establece como el directorio de trabajo. Finalmente, se ejecuta la aplicación con la línea `CMD ["python", "app.py"]`, lo que mantendrá activos los contenedores resultantes de dicha imagen.

```
# Se usa la imagen especifica del software como padre
FROM software_image

# Se copian los archivos de la aplicacion en la imagen
COPY . /var/www/html/app

# Se establece el directorio de trabajo
WORKDIR /var/www/html/app

# Se ejecuta la aplicacion
CMD ["python", "app.py"]
```

Listing 2.2: Dockerfile de una imagen secundaria (hija)

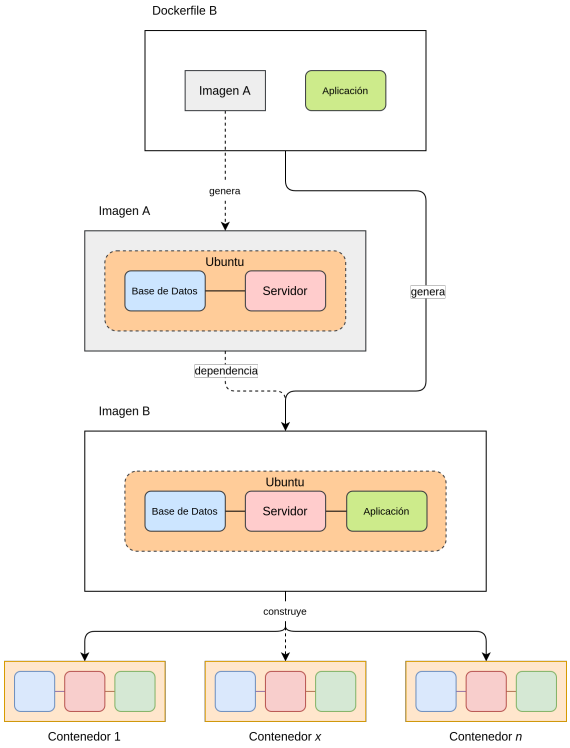


Figura 2.7: Creación de contenedores a partir de una imagen *A* padre

Solución del proyecto

Si bien el enfoque planteado por el artículo presenta una mayor eficiencia, se ha usado una solución más simple y minimalista para el proyecto, ya que los conceptos planteados en el catálogo (capítulo 4) no requieren de una gran cantidad de software para su funcionamiento, y por tanto, no es necesario crear un sistema de imagen padre e imágenes hijas.

La mayoría de laboratorios de este proyecto solo requieren de la instalación de herramientas, por lo que basta con usar una imagen de un sistema operativo y añadir las herramientas necesarias para cada laboratorio a través de sus ficheros Dokerfile. Se han usado dos imágenes de sistemas operativos basados en Linux para los laboratorios:

- **Alpine Linux** [19]: un sistema construido especialmente para su uso en contenedores, ya que es muy ligero, eficiente y seguro; se usó en aquellos laboratorios que no emplean herramientas de penetración convencionales.
- **Debian** [20]: uno de los sistemas más usados y reconocidos mundialmente; se usó en aquellos laboratorios que emplean herramientas de penetración habituales.

Las herramientas mencionadas se han obtenido a través de sus repositorios oficiales y mediante el gestor de paquetes de cada sistema operativo (`apk` para Alpine Linux y `apt` para Debian).

Por otra parte, existen laboratorios que emplean imágenes específicas que ya implementan un servicio o funcionalidad concreta con el que poder experimentar. Estas imágenes se han obtenido a través de Docker Hub y ofrecen la misma dinámica que el resto de laboratorios.

2.3. Análisis de tecnologías para la web

Esta sección, al contrario de lo que ocurre con la sección 1.5 donde se muestra un resumen de las tecnologías utilizadas, recoge los análisis realizados a las herramientas relacionadas con el desarrollo de la plataforma web, pues se ha considerado necesario profundizar más en esta parte del proyecto.

2.3.1. Página web

Características	WordPress	Astro	Drupal
Tipo de plataforma	CMS	Framework	CMS
Orientación	Sitios web: pequeños y medianos	Aplicaciones web	Sitios web: grandes y complejos
Lenguaje de programación	PHP	Javascript	PHP
Facilidad de uso	Simple: no requiere experiencia técnica ni programación	Compleja: requiere experiencia técnica y en programación	Compleja: requiere experiencia con el propio CMS
Personalización	Alta: temas y plugins	Muy alta: desarrollo web	Alta: módulos y extensiones
Escalabilidad	Sí: sitios web pequeños y medianos	Sí: aplicaciones web pequeñas y grandes	Sí: sitios grandes y complejos
Comunidad	Grande: usuarios y desarrolladores	Creciente: desarrolladores	Mediana: usuarios y desarrolladores, muy dedicados y comprometidos
Popularidad	El CMS más popular	Un framework relativamente nuevo	Uno de los CMS más usados
Seguridad	Alta: pero es vulnerable a ataques por su popularidad	Variable: depende de la implementación del desarrollador	Alta: CMS centrado en la seguridad y la protección

Tabla 2.1: Comparativa entre WordPress, Astro y Drupal

Se ha elegido WordPress como la plataforma para la creación de la página web del proyecto debido a sus ventajas en cuanto a facilidad de uso, personalización y escalabilidad, que eran los principales aspectos a tener en cuenta para el desarrollo.

WordPress y Astro

Facilidad de uso Factor importante en la decisión de elegir esta herramienta, ya que no se cuentan con conocimientos previos de programación web a nivel técnico, por lo que debía usarse una herramienta que no tuviera en cuenta esas necesidades.

WordPress parece cubrir ese aspecto de forma notable debido a que su popularidad se basa, precisamente, en la cantidad de usuarios que son capaces de crear una página web sin necesidad de ser desarrolladores o contar con experiencia previa.

En cambio, Astro [21] es un framework para desarrollo de aplicaciones web en lugar de un CMS, por lo que inicialmente requiere de cierta experiencia previa en programación web para poder usarlo.

Personalización También fue considerada un factor clave ya que se contaba con la creación de una página web atractiva y funcional.

Teniendo en cuenta lo mencionado en el punto anterior acerca de la falta de conocimientos previos, se hubiera empleado una gran cantidad de tiempo no solo en aprender a elaborar un buen *front-end* desde cero, sino que el objetivo principal de este trabajo de fin de grado no se centra tanto en la página web, sino en el sistema completo, pudiendo considerar la mejora del *front-end* como una posible futura línea de trabajo (sección 5.3).

Escalabilidad Uno de los objetivos (sección 1.2) a tener en cuenta para la plataforma, era que esta fuera fácilmente escalable.

La escalabilidad de WordPress es una de sus principales fortalezas y uno de los motivos por los que es una plataforma popular y confiable para la creación de sitios web de cualquier tamaño. Su arquitectura modular y su capacidad para trabajar con bases de datos y servidores de alta gama, la convierten en una opción sólida para aquellos que buscan crear sitios web que puedan crecer y adaptarse a sus necesidades en el futuro.

Astro no se queda atrás, ya que también es una herramienta escalable, pero en este caso, para el desarrollo de aplicaciones web. No obstante, aunque uno de los pilares de esta herramienta sea precisamente, la escalabilidad, no deja de ser un factor variable que depende de la implementación del desarrollador, por lo que no se considera una opción tan sólida como WordPress.

WordPress y Drupal

También se tuvo en cuenta Drupal [22], otro CMS *open-source* que se encuentra entre los más populares y que, a pesar de no ser tan conocido como WordPress, también es una opción muy interesante para la creación de páginas web.

Facilidad de uso, personalización y escalabilidad Observando la tabla anterior se puede apreciar varios motivos por los que se ha optado usar Wordpress en lugar de Drupal, pero todos ellos relacionados al mismo punto: la complejidad de Drupal no se ajusta a las necesidades de este proyecto.

Si bien puede ser una idea interesante a largo plazo y en un hipotético caso de evolución de la plataforma web a un proyecto superior, pero para el propósito actual, se considera óptimo elegir una herramienta más sencilla y fácil de usar.

2.3.2. Base de Datos

SQLite vs MySQL

SQLite [23] es un RDBMS que se caracteriza por ser ligero, rápido y fácil de usar. Se trata de una buena opción para proyectos pequeños, pero no es recomendable para proyectos de gran envergadura, ya que no es capaz de manejar grandes cantidades de datos.

Por otro lado, MySQL [24] también es otro RDBMS que se caracteriza por ser rápido, seguro y fácil de usar. Al contrario que con SQLite, MySQL sí es recomendable para proyectos de gran envergadura, ya que es capaz de manejar grandes cantidades de datos.

Características	SQLite	MySQL
Tipo de base de datos	Relacional, integrada	Relacional, cliente-servidor
Gestión de usuarios	Debe programarse	Integrada
Escalabilidad	Limitada: por su naturaleza integrada	Escalable: en función del hardware disponible
Confiabilidad	Menor capacidad de recuperación de datos	Mayor capacidad de recuperación de datos
Flexibilidad	Limitada en cuanto a configuración de memoria	Mayor flexibilidad en configuración
Seguridad	Limitada: no ofrece encriptación de datos	Mayor seguridad: ofrece encriptación de datos

Tabla 2.2: Comparativa entre SQLite y MySQL

En función de la naturaleza del proyecto a realizar, se ha considerado usar MySQl debido a las siguientes razones:

Gestión de usuarios integrada Al tratarse de una plataforma web que requiere la gestión de usuarios, MySQL ofrece una gestión de usuarios integrada, lo que simplifica y agiliza el proceso de gestión de usuarios en comparación con SQLite.

Escalabilidad MySQL ofrece una mayor escalabilidad que SQLite debido a su arquitectura cliente-servidor. Esto significa que puede manejar grandes cantidades de datos y muchos usuarios de manera simultánea, lo que es importante en un entorno en el que se espera que varios usuarios se conecten y utilicen la plataforma al mismo tiempo.

Confiabilidad MySQL ofrece una mayor capacidad de recuperación de datos en comparación con SQLite. Esto es importante en un proyecto que busca ser utilizado por varias personas, ya que cualquier pérdida de datos podría tener un impacto significativo.

Seguridad MySQL ofrece mayor seguridad que SQLite al contar con encriptación de datos. Esto es importante en un proyecto que busca proteger los datos de los usuarios y prevenir posibles ciberataques.

Además, también se ha tenido en cuenta su integración con WordPress, herramienta seleccionada para la página web de la plataforma, ya que WordPress cuenta con integración inmediata con MySQL desde su instalación.

2.3.3. Alojamiento

Local

Inicialmente, se planteó el uso de un servidor local para el alojamiento de la plataforma web, lo que permite realizar pruebas de forma maś rápida y sencilla, sin necesidad de contratar un servidor externo.

Linode

Esta empresa ofrece servicios de alojamiento en la nube mediante servidores virtuales privados (VPS) con diferentes características y precios, lo que permite a los usuarios elegir el plan que mejor se adapte a sus necesidades.

Además, Linode cuenta con una interfaz de usuario intuitiva y fácil de usar, así como con una amplia documentación y soporte técnico para ayudar a los usuarios a configurar y administrar sus servidores.

Este proyecto requiere la gestión de contenedores Docker, por lo que se buscaba un servicio que integrara una solución similar a Kubernetes para facilitar la gestión de los mismos, y Linode ofrece un servicio que permite a los usuarios implementar y administrar clústeres de Kubernetes en la nube: LKE.

Linode Kubernetes Engine (LKE) Este servicio se encarga de la configuración, el aprovisionamiento y la administración de los nodos de un clúster, lo que permite a los usuarios centrarse en el desarrollo de sus aplicaciones en lugar de en la gestión de la infraestructura subyacente.

Además, Linode Kubernetes Engine (LKE) es compatible con herramientas y servicios de terceros, lo que permite a los usuarios integrar fácilmente sus aplicaciones con otros servicios en la nube.

Amazon Web Service (AWS)

Esta plataforma de servicios en la nube ofrece una amplia gama de servicios, incluyendo almacenamiento, bases de datos, redes, etc.

AWS permite a los usuarios crear aplicaciones escalables y de alta disponibilidad sin necesidad de invertir en hardware o infraestructura física.

Como se mencionó anteriormente, este proyecto requiere la gestión de contenedores Docker, por lo que se buscaba un servicio que integrara Kubernetes para facilitar la gestión de los mismos. AWS cuenta con el servicio Amazon Elastic Kubernetes Service (EKS), que permite a los usuarios implementar y administrar clústeres de Kubernetes en la nube.

Amazon Elastic Kubernetes Service (EKS) Este servicio se encarga de la configuración, el aprovisionamiento y la administración de los nodos del clúster, lo que permite a los usuarios centrarse en el desarrollo de sus aplicaciones en lugar de en la gestión de la infraestructura subyacente.

Google Cloud Platform (GCP)

Esta plataforma de Google ofrece una gran cantidad de servicios en la nube con la que poder alojar distintos proyectos en función de sus necesidades: almacenamiento, bases de datos, redes, análisis de datos, aprendizaje automático, etc.

Se planteó el uso de esta herramienta para el alojamiento del proyecto porque ofrece un saldo gratuito equivalentes a 3 meses de uso de sus servicios, resultando muy interesante para el desarrollo del mismo.

Cuenta con varios servicios potencialmente compatibles con el proyecto, de los cuales se analizaron los siguientes:

Google Kubernetes Engine (GKE) Este servicio permite ejecutar aplicaciones en contenedores de forma sencilla y rápida, sin necesidad de gestionar la infraestructura subyacente. Se planteó como una opción para la gestión de los contenedores Docker que hacen de laboratorios del proyecto.

Sin embargo, se descartó esta opción debido a que la complejidad de la herramienta no se ajustaba a las necesidades del proyecto, ya que se buscaba una herramienta que facilitara la gestión de los contenedores Docker, pero de forma más sencilla.

Google Cloud Run Este servicio de computación sin servidor (*serverless*) de Google permite a los desarrolladores implementar fácilmente aplicaciones en contenedores en un entorno completamente administrado, sin la necesidad de administrar la infraestructura subyacente.

Cloud Run es compatible con contenedores basados en Docker, lo que significa que es posible empaquetar una aplicación en un contenedor y luego implementarla en Cloud Run. Se planteó como una opción para el alojamiento completo del proyecto, pero se descartó tras analizarlo, ya que este no requiere de un servicio de computación sin servidor.

Por otra parte, como se mencionó, esta herramienta está más orientada al uso de aplicaciones en contenedores, mientras que este proyecto se basa en la gestión de contenedores Docker en sí misma.

Google Compute Engine Este servicio de infraestructura (IaaS) de Google permite crear máquinas virtuales en la nube, lo que resulta muy interesante para el desarrollo de este proyecto.

Compute Engine permite a los usuarios tener un control total sobre la configuración y personalización de sus máquinas virtuales: cantidad de CPU, memoria RAM, almacenamiento y la opción de elegir entre diferentes tipos de instancias según el rendimiento y el costo requeridos. También es posible seleccionar el sistema operativo y personalizar la configuración de red y seguridad de las instancias.

Este servicio se considera el más compatible con la arquitectura del proyecto, ya que en términos simples, podría implementarse en un primer momento de forma local, y luego trasladarlo a una instancia de Compute Engine sin necesidad de realizar grandes cambios.

Esta acción intentó llevarse a cabo una vez finalizado el desarrollo de la plataforma web, pero no pudo aplicarse adecuadamente debido a los continuos problemas de visibilidad que presentaba la instancia generada en Compute Engine, impidiendo el acceso público a la plataforma web. Tuvo que descartarse la idea en base al tiempo restante del proyecto.

CAPÍTULO 3

Diseño y desarrollo de la plataforma

3.1. Ingeniería de Requisitos

El único actor de la plataforma será el usuario que haga uso de ella, esté o no registrado en la misma, por lo que no se especificará este detalle en los Casos de Uso (sección 3.1.1) puesto que siempre será el mismo.

3.1.1. Casos de uso

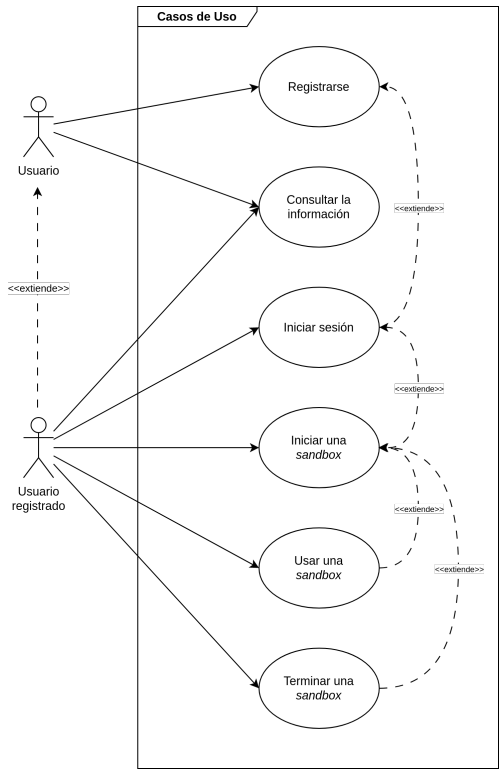


Figura 3.1: Casos de uso

3.1.2. Requisitos funcionales

Los casos de uso permiten identificar los requisitos funcionales necesarios para satisfacer las necesidades del usuario y asegurarse de que el sistema cumpla con sus objetivos y expectativas. Teniendo eso en cuenta, los casos de uso describen cómo interactúan los usuarios con el sistema y qué acciones o funcionalidades deben estar disponibles en cada situación.

Para este proyecto, se han definido los siguientes requisitos en función de los casos de uso mencionados en el apartado anterior:

Usuarios

Requisitos relacionados con la gestión de usuarios y sus datos.

RF - 01	**Registro de usuarios**
El sistema debe permitir que los usuarios se registren y creen una cuenta en la plataforma para poder usar los laboratorios de pruebas.	
▪ Nombre de usuario. ▪ Dirección de correo electrónico. ▪ Contraseña.	

RF - 02	**Inicio de sesión**
El sistema debe permitir que los usuarios registrados puedan iniciar sesión para poder acceder al uso de los laboratorios de pruebas.	

RF - 03	**Validación de los datos de registro**
El sistema debe comprobar que los datos de registro de un usuario sean válidos.	
▪ No se registró un usuario con el mismo nombre de usuario ni correo electrónico. ▪ Si alguno de los datos no es válido (error), se informará al usuario.	

Contenido

Requisitos relacionados con la gestión de contenido y sus características.

RF - 04	**Consulta de contenido**
El sistema debe proporcionar contenido relevante que permita a los usuarios adquirir los conocimientos necesarios para realizar las pruebas.	

Laboratorios de pruebas

Requisitos relacionados con la gestión de los entornos virtualizados.

RF - 05	**Inicio de laboratorios de pruebas**
El sistema debe permitir que los usuarios seleccionen e inicien un laboratorio de pruebas que deseen realizar.	
▪ Si no es posible iniciar un laboratorio (error), se informará al usuario.	

RF - 06	**Tiempo de vida de los laboratorios de pruebas**
El sistema debe destruir automáticamente un laboratorio iniciado una vez que se haya superado un tiempo de vida definido.	
▪ Definir un tiempo de vida fijo para los laboratorios de pruebas.	

RF - 07	**Número máximo de laboratorios de pruebas**
El sistema no debe permitir el inicio de infinitos laboratorios por parte de un usuario.	
▪ Un usuario solo podrá tener activo un laboratorio a la vez.	

RF - 08	**Uso de un laboratorio de pruebas iniciado**
El sistema debe permitir que los usuarios puedan conectarse a un laboratorio una vez este se haya iniciado.	
▪ Proporcionar los datos de conexión a un laboratorio al usuario. ▪ La conexión se realizará por el propio usuario a través de SSH.	

RF - 09	**Detención de laboratorios de pruebas**
El sistema debe permitir que los usuarios puedan apagar un laboratorio de pruebas, lo que equivale a destruirla manualmente (en lugar de esperar a que se cumpla su tiempo de vida).	
▪ Si no es posible destruir un laboratorio (error), se informará al usuario. ▪ Un administrador podrá acceder a registros de uso de los laboratorios.	

3.1.3. Requisitos no funcionales

Por otro lado, los requisitos no funcionales describen las cualidades o atributos que debe tener un sistema, enfocándose en cómo hace el sistema lo que hace; es decir, cómo se comporta en términos de calidad.

Para este proyecto, se han definido los siguientes requisitos en función de los requisitos mencionados en la sección anterior, así como los objetivos de la plataforma:

RNF - 01	**Privacidad de los datos**
El sistema debe garantizar la seguridad y privacidad de los datos de los usuarios.	
▪ Aplicar una política de cifrado para las contraseñas.	

RNF - 02	**Aislamiento de los laboratorios de pruebas**
El sistema debe garantizar la seguridad y aislamiento de los laboratorios de prueba de la plataforma.	
▪ Aislar completamente los laboratorios sin afectar al resto de la plataforma.	

RNF - 03	**Seguridad de la plataforma**
El sistema debe implementar medidas de seguridad para prevenir ataques externos o internos a la plataforma.	
▪ Medidas de autenticación y autorización para garantizar el acceso solo a usuarios autorizados.	

RNF - 04	**Accesibilidad**
La plataforma debe ser accesible para cualquier usuario, independientemente de su nivel de experiencia en ciberseguridad.	
▪ Contenido claro, accesible y bien estructurado.	

RNF - 05	**Usabilidad**
La plataforma debe ser fácil de usar y accesible para cualquier usuario, independientemente de su nivel de experiencia en ciberseguridad.	
▪ Interfaz de usuario intuitiva, clara y organizada.	

RNF - 06	**Extensibilidad**
La plataforma debe ser fácilmente extensible, permitiendo la futura integración de más contenido y laboratorios de pruebas.	
▪ Definir y documentar claramente cómo añadir contenido y laboratorios.	

3.2. Arquitectura

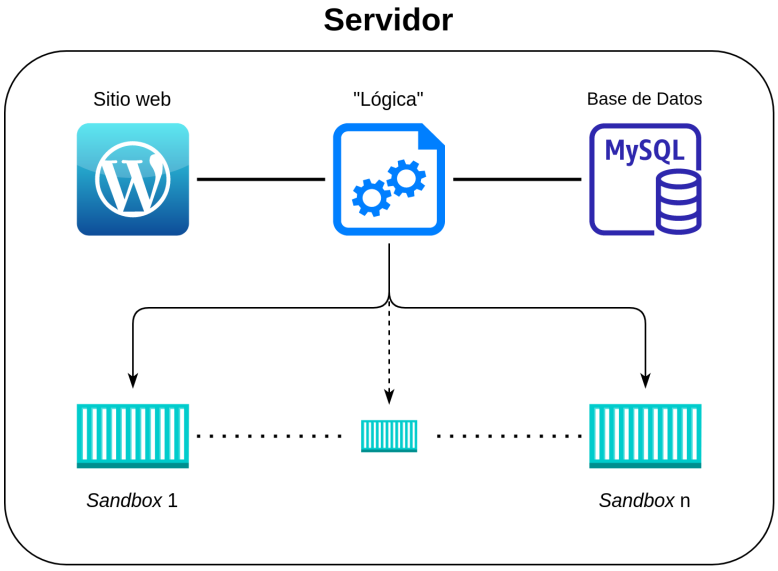


Figura 3.2: Arquitectura

Los componentes de la plataforma podrían dividirse en 4 partes fundamentales: el *front-end*, el *back-end*, la base de datos y los contenedores Docker (laboratorios). Todos estos están desplegados en un servidor que utiliza LAMP.

3.2.1. *Front-end*: WordPress

El *front-end* es la parte de la plataforma que interactúa con el usuario; en este caso, se ha desarrollado una aplicación web que permite al usuario acceder a la plataforma y realizar las acciones que se describen en la sección 3.1.1.

Se ha usado un tema hijo generado a partir del tema Divi, definido como *divi-child*; dicho tema actua como eje central tanto para el diseño como para la funcionalidad de la plataforma web, como se describe en el siguiente apartado.

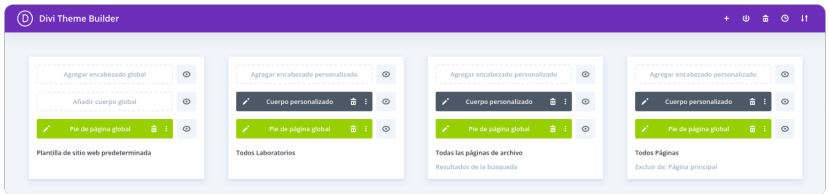


Figura 3.3: Estructura de la plataforma web

3.2.2. *Back-end*: `functions.php`

El *back-end* se encarga de gestionar las peticiones del usuario y de comunicarse con la base de datos y los contenedores Docker. Se ha usado el fichero `functions.php` del tema *divi-child* para implementar el *back-end* de la plataforma, ya que este fichero permite extender las funcionalidades de un sitio web de WordPress de forma sencilla.

Este fichero es propio de cada tema, y se ejecuta en cada petición que se realiza a la plataforma, lo que permite interactuar con la página al mismo tiempo que lo hace un usuario.

3.2.3. *Back-end*: tareas de Cron

Las tareas de cron son acciones que se ejecutan de forma periódica en un sistema, definidas como instrucciones en un fichero de configuración. Se ha usado una para comprobar si algún laboratorio de pruebas superó su tiempo de vida (1 hora) y detenerlo de forma automática.

```
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

* * * * * /bin/bash /var/www/html/scripts/stop-1h-container.sh > /var/www/html/
    scripts/stop-1h-container.log 2>&1
```

Listing 3.1: contenido del fichero

Esta tarea se ha implementado directamente en el servidor, ya que se consideró la opción más rápida y sencilla: se ejecuta cada minuto y destruye aquellos contenedores -laboratorios- que hayan excedido su tiempo de vida, liberando recursos en el sistema y permitiendo un uso de contenedores dinámico.

3.2.4. Base de datos: MySQL

WordPress gestiona todo su contenido a través de una base de datos: configuración, publicaciones, usuarios registrados... aprovechando ese funcionamiento, se ha usado la base de datos de WordPress para almacenar los datos de los laboratorios de pruebas, y concretamente, se ha usado la tabla `wp_usermeta`.

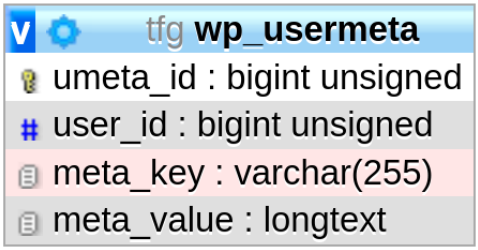


Figura 3.4: Estructura de la tabla `wp_usermeta`

El fichero `functions.php` conecta con la base de datos cada vez que el usuario quiere interactuar con un laboratorio desde la plataforma, y usa la tabla `wp_usermeta` para almacenar los datos de los laboratorios en uso (el usuario que lo está usando, el puerto de conexión, etc.).

	umeta_id	user_id	meta_key	meta_value
Editar Copiar Borrar	68	1	metaboxhidden_dashboard	a:0:{}
Editar Copiar Borrar	234	1	session_tokens	a:4:{s:64:"e7b32f64cf4537557a65d679598154b72f34fce...
Editar Copiar Borrar	248	1	wpcf7_hide_welcome_panel_on	a:1:{i:0;s:3:"5.7";}
Editar Copiar Borrar	249	1	_new_email	a:2:{s:4:"hash";s:32:"14f2b3214188e73317f5f99b7845...
Editar Copiar Borrar	250	1	_current_lab_info_	{"ip":"127.0.0.1","port":7540,"date":{"date":"2023...

Figura 3.5: Datos de la tabla `wp_usermeta`

- **user_id**: identificador del usuario que ha realizado la acción; en este caso, el usuario que está usando un laboratorio de pruebas.
- **meta_key**: identificador del tipo de valor que se está almacenando; en este caso, información de un laboratorio activo.
- **meta_value**: información que se almacena; en este caso, los datos de un laboratorio activo (ID del contenedor, puerto...) en formato JSON.

A continuación se muestra el contenido del laboratorio activo en la figura 3.5:


```
{
    "ip": "127.0.0.1",
    "port": 7540,
    "date": {
        "date": "2023-09-06 15:26:16.615559",
        "timezone_type": 3,
        "timezone": "UTC"
    },
    "post_id": 79,
    "id": "b5f344fd6d2f3c8796d421fb6274c3b7d984fdeebbec2755e531dee11570879c"
}
```


3.2.5. Contenedores Docker: laboratorios de pruebas

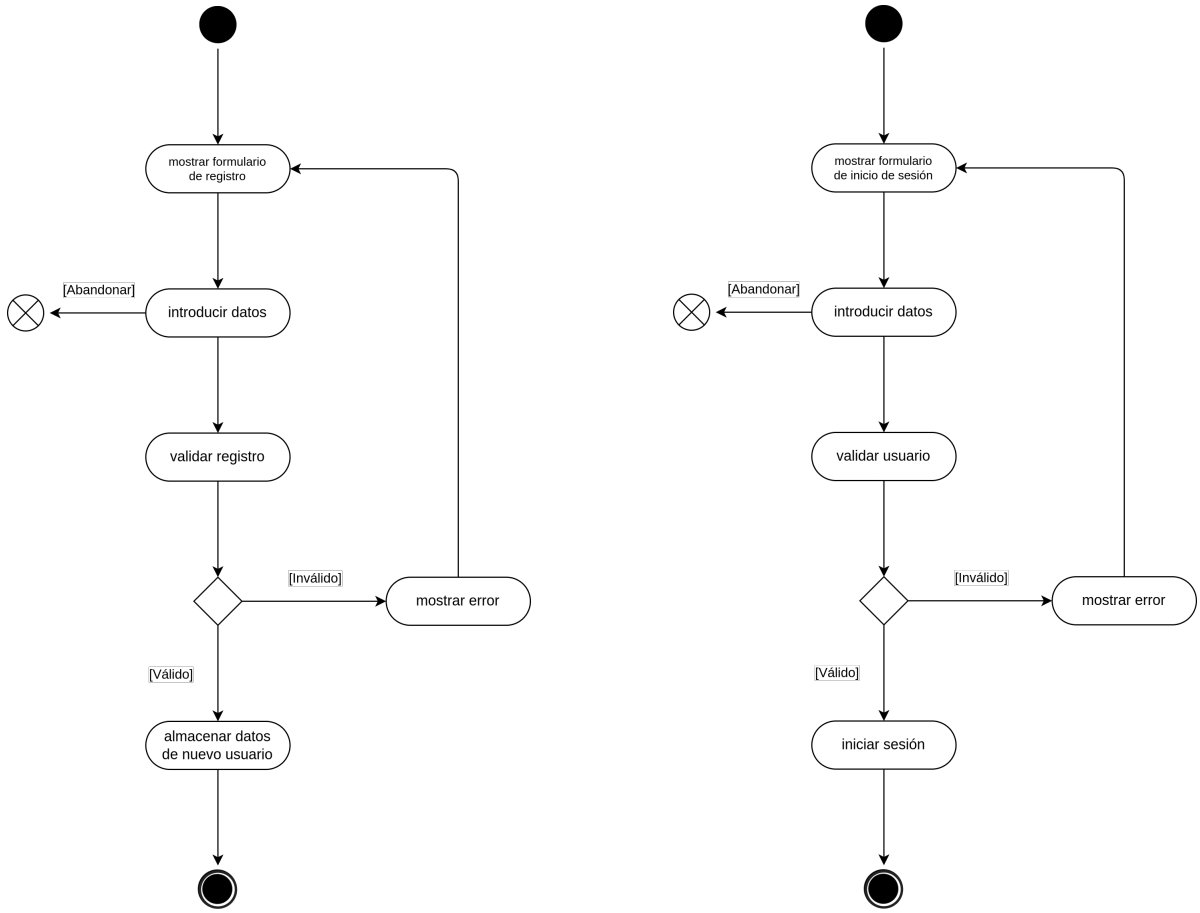
Los contenedores Docker son la parte de la plataforma que se encarga de ejecutar los laboratorios de pruebas. En este caso, se ha usado para ejecutar los laboratorios de pruebas que han sido creados por los usuarios.

Los laboratorios son construidos a partir de imágenes contruidas con ficheros `Dockerfile` locales, por lo que pueden crearse nuevos laboratorios de pruebas de forma rápida y sencilla para extender la plataforma. Esto se explica más detalladamente en los apéndices del documento.

3.3. Modelado de actividades y transiciones

Los diagramas mostrados a continuación presentan las actividades que puede realizar un usuario en la plataforma y las transiciones entre dichas actividades.

3.3.1. Tratamiento de usuarios



(a) Registro de un usuario

(b) Inicio de sesión de un usuario

Figura 3.6: Modelado de actividades y transiciones

Los diagramas presentados en esta sección definen la gestión de usuarios que permite el registro e inicio de sesión de los mismos en la plataforma.

El proceso de registro de un usuario mostrará inicialmente un formulario para poder obtener sus datos, debiendo verificar que dichos datos no pertenecen a un usario previamente registrado, puesto que los usuarios deben ser únicos.

El proceso de inicio de sesión de un usuario mostrará un funcionamiento similar al de registro, pero esta vez, para comprobar que el usuario sí ha sido registrado anteriormente.

3.3.2. Consulta de documentación

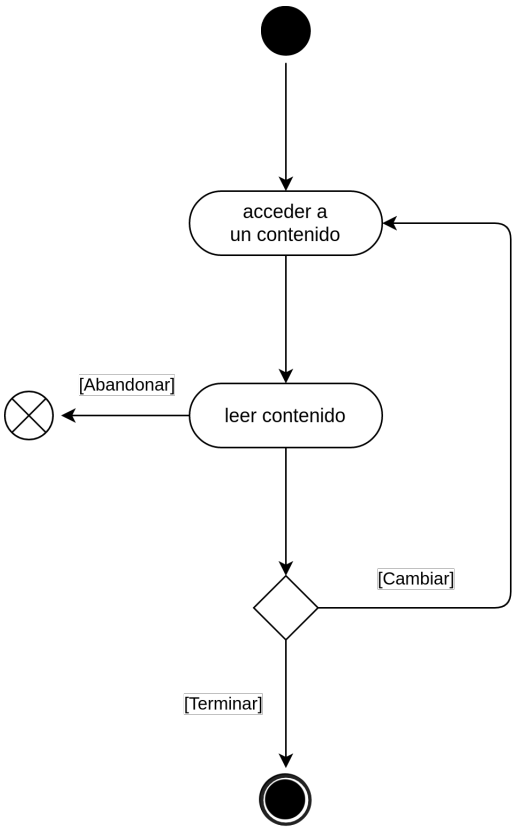
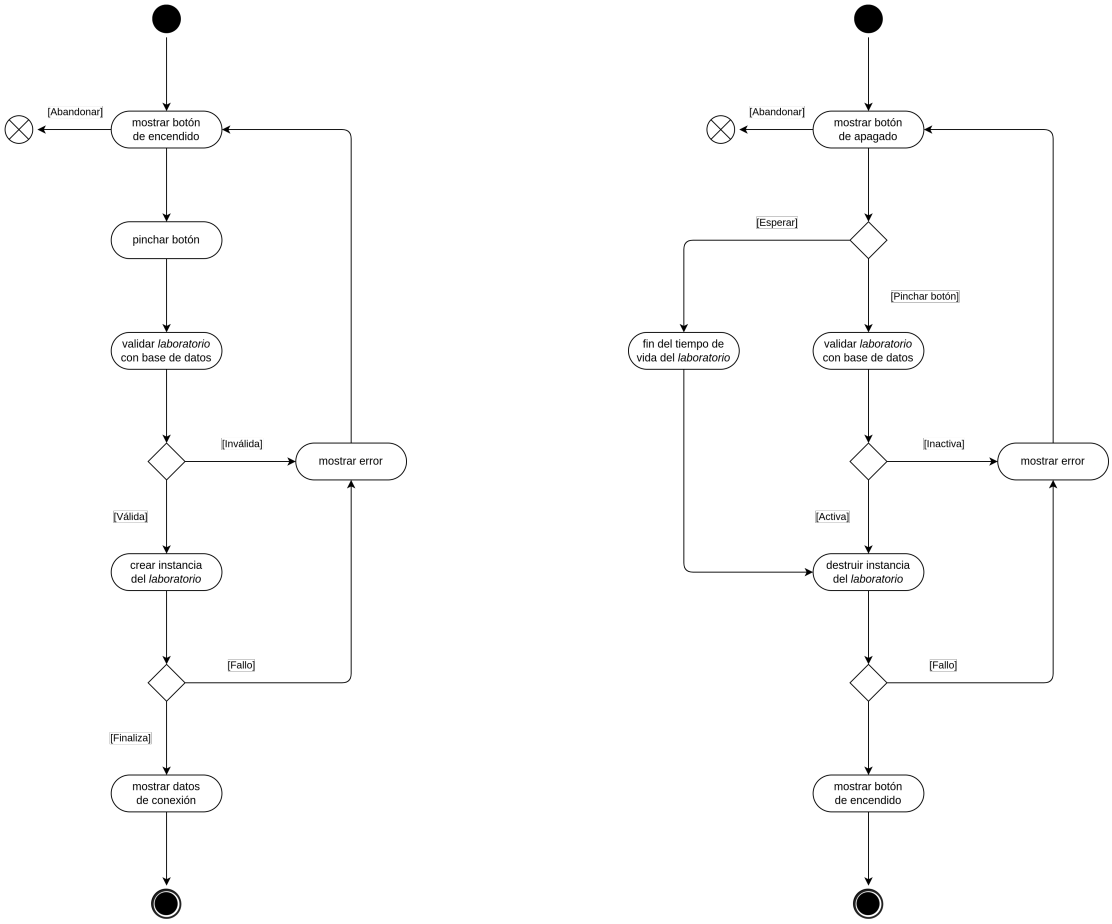


Figura 3.7: Consulta de documentación

El diagrama presentado en esta sección define el consumo de contenido de la plataforma por parte de un usuario, esté registrado o no, ya que el contenido instructivo de la plataforma se considera público, al contrario que el uso de los laboratorios que requerirá un registro por parte del usuario.

El proceso de consulta de documentación es bastante simple: un usuario puede consumir contenido de forma continua, pero al estar dividido por conceptos, necesitará cambiar de ubicación dentro de la plataforma para poder seguir accediendo a contenido nuevo.

3.3.3. Tratamiento de laboratorios



(a) Inicio de un laboratorio

(b) Detención de un laboratorio

Figura 3.8: Tratamiento de laboratorios

Los diagramas presentados en esta sección definen la gestión de entornos virtualizados de la plataforma.

El proceso de inicio de un laboratorio iniciará pulsando un botón, preferiblemente ubicado al final de un contenido instructivo descrito anteriormente en la sección 3.3.2, debiendo verificar que es posible crear dicho laboratorio.

El proceso de detención de un laboratorio seguirá el mismo proceso que el anterior, pero realizará la opción opuesta.

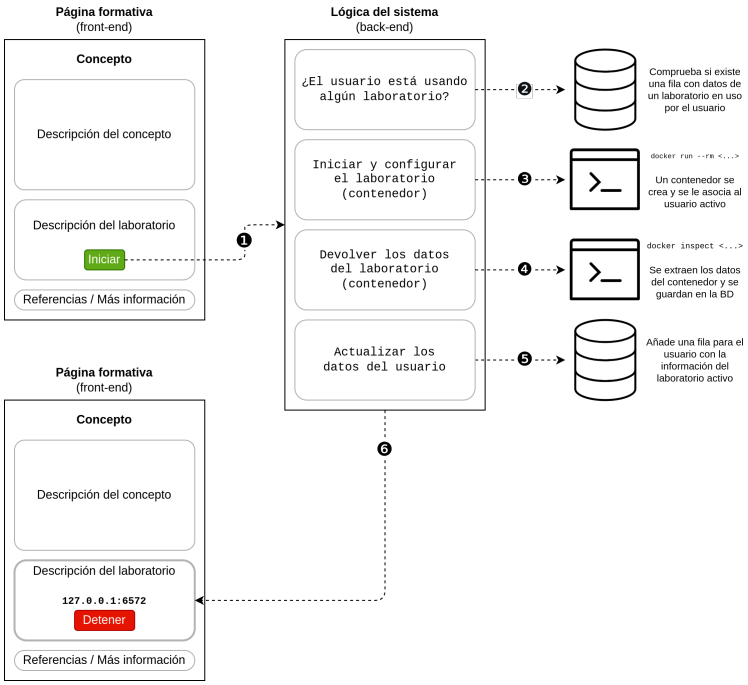


Figura 3.9: Inicio de un laboratorio desde el sitio web

Este diagrama representa un uso normal de la plataforma, donde el usuario accede a la página web y desea iniciar un laboratorio para probar el concepto tratado en la página informativa.

Se detallan a continuación los pasos que se llevan a cabo:

1. El usuario solicita iniciar un laboratorio para probar el concepto tratado en la página informativa pulsando el botón de inicio, ejecutando algunas de las funcionalidades de la plataforma definidas en `functions.php`.
2. Se comprueba en la base de datos que el usuario activo no tiene ningún otro laboratorio activo. Si lo tuviera, mostraría una advertencia y un enlace a la página del laboratorio activo en la sección *Laboratorio* de la página.
3. Se obtiene un puerto disponible en el servidor y se hace una llamada al sistema para crear un contenedor Docker mapeando su puerto 22 (SSH por defecto) con el puerto obtenido. La ejecución del comando en el sistema devuelve la ID del contenedor creado, almacenándola en `functions.php`.
4. Se vuelve a llamar al sistema para extraer los datos de dicho contenedor usando el ID del paso anterior, porque en el paso anterior el contenedor todavía no existía y por tanto, no era posible extraer información.
5. Se registran los datos recopilados en la base de datos en formato JSON para usar una única celda de la tabla y poder extraerlos cómodamente en `functions.php`.
6. Se cambia el estado de la sección *Laboratorio* de la página: se añade la IP de la plataforma (en este caso, `localhost`) y el puerto del sistema asociado al contenedor, y se cambia el botón de la sección para que ahora detenga el laboratorio.

Para facilitar la interpretación, no se han contemplado los errores que pueden ocurrir durante el proceso.

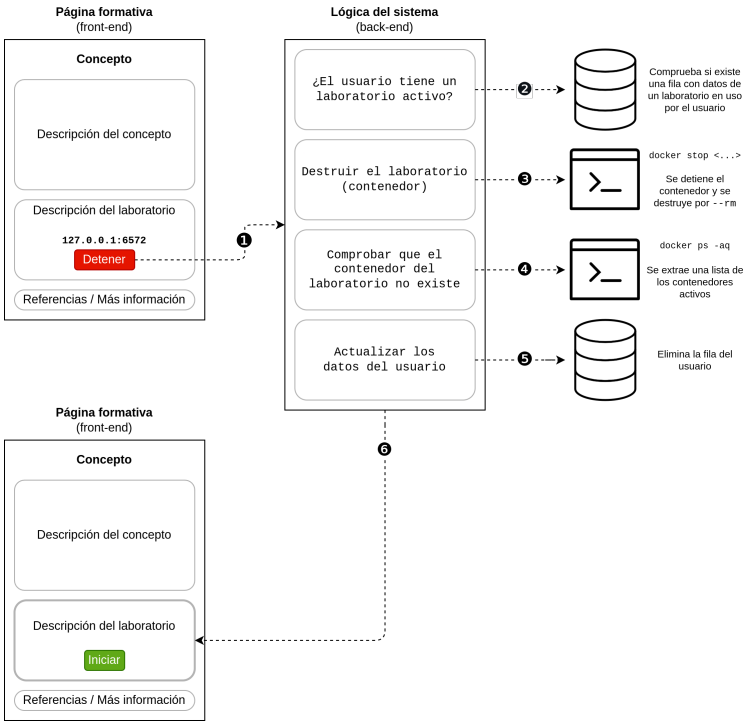


Figura 3.10: Detención de un laboratorio desde el sitio web

Este diagrama representa un uso normal de la plataforma, donde el usuario accede a la página web y desea detener el laboratorio activo.

Se detallan a continuación los pasos que se llevan a cabo:

1. El usuario solicita detener el laboratorio activo porque ha terminado de usarlo (o quiere usar otro) pulsando el botón de detener, ejecutando algunas de las funcionalidades de la plataforma definidas en `functions.php`.
2. Se comprueba en la base de datos que el usuario activo tiene un laboratorio activo y se obtienen los datos del mismo.
3. Se hace una llamada al sistema para detener el contenedor Docker usando la ID obtenida en el paso anterior, y como todos los contenedores se han creado con la opción `--rm`, al detenerse son eliminados.
4. Se hace una llamada al sistema para comprobar que el contenedor se ha eliminado correctamente. Se ejecuta el comando `docker ps -aq` (lista de IDs de todos los contenedores existentes) y se comprueba que no aparece el contenedor en la lista.
5. Se elimina la entrada de la base de datos asociada al laboratorio activo.
6. Se cambia el estado de la sección *Laboratorio* de la página: se eliminan la IP y el puerto asociados y se cambia el botón de la sección para que ahora inicie de nuevo el laboratorio.

Para facilitar la interpretación, no se han contemplado los errores que pueden ocurrir durante el proceso.

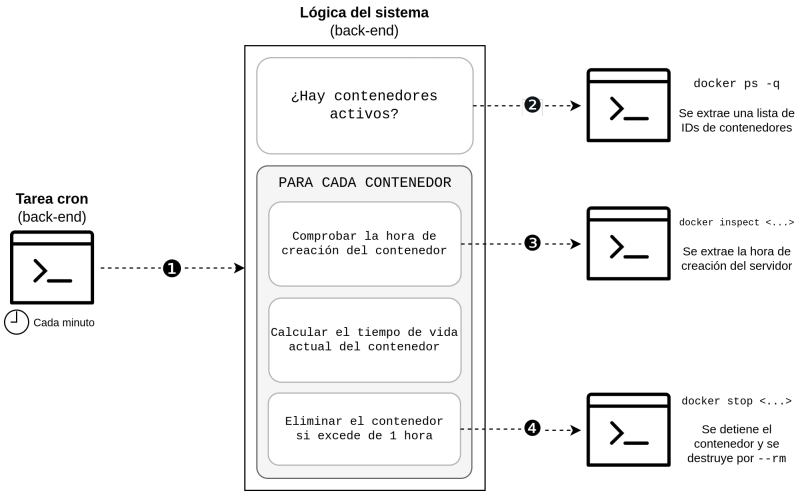


Figura 3.11: Destrucción automática de un laboratorio

Este diagrama representa un uso normal de una tarea de cron con la que detener y eliminar automáticamente laboratorios que excedan su tiempo de vida.

Se detallan a continuación los pasos que se llevan a cabo:

1. La tarea de cron del sistema ejecuta un script de Bash (apéndice C.1) para detener los contenedores Docker que se encuentren activos y cuyo tiempo de vida exceda a 1 hora. Esta tarea se ejecuta cada minuto.
2. Se hace una llamada al sistema para comprobar si hay contenedores activos y en caso afirmativo, obtener una lista de los IDs de esos contenedores. El script termina en caso de no haber ninguno.
3. Se usa la lista en un bucle para iterar sobre ella y se hace una llamada al sistema para obtener la hora de creación del contenedor actual. Esos datos se usan para calcular el tiempo de vida del contenedor actual.
4. Se comprueba si el tiempo de vida del contenedor actual excede a 1 hora y si es así, se detiene. Como los contenedores han sido creados con la opción `--rm`, al detenerse son eliminados.

Este script muestra información cada vez que se ejecuta, haciendo que la tarea de cron almacene dicha información en un archivo de texto a modo de registro.

```
Ejecución actual: 13:14:06

Hay 4 contenedores activos.

592cd7fed0cc:   903 / 3600 segundos. Disponible
d37127f27ae4:   3000 / 3600 segundos. Disponible
6159264a6115:   3545 / 3600 segundos. Destrucción inminente
31e11305ce36:   3610 / 3600 segundos. Tiempo de vida alcanzado
```

Figura 3.12: Salida del script de la tarea de cron

CAPÍTULO 4

Catálogo de conceptos

A continuación se muestra una serie de conceptos relacionados con el pentesting. El catálogo incluye algunos conceptos más genéricos y comunes, así como casos más específicos, todo con el fin de ofrecer al usuario una visión general del pentesting.

Cada concepto de este capítulo incluye una pequeña descripción, el motivo de su inclusión al sitio web y un resumen de la funcionalidad de su laboratorio asociado.

Por último, cabe destacar que el texto mostrado en este documento no se corresponde con la versión final -más elaborada y detallada- publicada en el sitio web del proyecto, ya que está orientado a la redacción de este trabajo de fin de grado y no al entorno en producción visible por el usuario de la plataforma.

4.1. Introducción a Linux

Linux es un sistema operativo de código abierto basado en UNIX creado en 1991, y aunque inicialmente fue desarrollado para ordenadores personales, actualmente se utiliza en una gran variedad de dispositivos (servidores, dispositivos móviles, sistemas embebidos, etc.).

Este concepto se presenta como una introducción al sistema de jerarquía de ficheros y directorios de Linux, así como a la gestión de permisos, grupos y usuarios del sistema. También se incluyó una introducción a la terminal de Linux y a distintos comandos clasificados por categorías.

4.1.1. Motivación

Este trabajo de fin de grado está orientado a aquellos usuarios que quieren empezar en el sector del pentesting, por lo que se ha considerado adecuado incluir una breve introducción a Linux, ya que resulta evidente el abundante uso de sistemas operativos basados en UNIX en el sector.

Tener un profundo conocimiento de la jerarquía de ficheros y directorios, la gestión de permisos, grupos y usuarios, así como el uso de la terminal, es fundamental para llevar a cabo pruebas de penetración efectivas y descubrir posibles vulnerabilidades y

puntos débiles en un sistema. Un pentester debe comprender esta jerarquía para ubicar rápidamente recursos críticos y datos sensibles.

Comprender cómo funcionan los permisos, cómo se asignan a usuarios y grupos específicos, y cómo se heredan en la jerarquía de archivos, es esencial para evaluar la seguridad. Los pentesters pueden identificar vulnerabilidades al explorar archivos con permisos incorrectos, que podrían permitir el acceso no autorizado o la ejecución de comandos peligrosos.

4.1.2. Laboratorio

Se proporciona al usuario un entorno virtualizado Linux donde poner en práctica lo mencionado anteriormente, revisar y manipular su contenido, así como ejecutar comandos.

Este entorno está pensado, principalmente, para aquellos usuarios que no dispongan de un sistema operativo basado en UNIX o que prefieran no modificar el suyo propio para realizar pruebas.

4.2. Introducción a Bash

Bash (*Bourne Again Shell*) es un lenguaje de scripting que se ejecuta en la terminal de Linux, y que permite automatizar tareas de forma sencilla.

Este concepto se presenta como una introducción al lenguaje Bash, abarcando desde la sintaxis básica hasta el uso de variables, funciones, bucles y condicionales.

4.2.1. Motivación

Bash es un lenguaje de scripting muy popular y ampliamente utilizado en el sector, ya que viene incluido en la mayoría de distribuciones de Linux y es el intérprete de comandos por defecto en la mayoría de ellas.

Durante una prueba de penetración, hay muchas tareas que se deben realizar de manera repetitiva, como el escaneo de puertos, la recolección de información, la enumeración de servicios, entre otros; es posible crear scripts de Bash que realicen estas tareas automáticamente, lo que ahorra tiempo y reduce errores humanos.

Resulta común crear exploits o pruebas específicas para aprovechar vulnerabilidades y evaluar la resistencia de un sistema durante una prueba de penetración, por lo que es importante contar con conocimientos de Bash para adaptarse a las necesidades de la prueba y crear scripts personalizados que permitan realizar pruebas de manera eficiente.

Por último, el proceso de crear scripts en Bash puede fomentar un conocimiento más profundo sobre cómo funcionan los sistemas operativos, lo que puede mejorar la comprensión general de los conceptos de seguridad informática y permitir al pentester detectar vulnerabilidades que podrían pasar desapercibidas de otra forma.

4.2.2. Laboratorio

Se proporciona al usuario un entorno virtualizado Linux donde poner en práctica lo mencionado anteriormente, revisar y manipular su contenido, así como ejecutar comandos.

Este entorno, al igual que el laboratorio anterior, está pensado para aquellos usuarios que no dispongan de un sistema operativo basado en UNIX o que prefieran no modificar el suyo propio para realizar pruebas.

4.3. Introducción a las redes

Una red de ordenadores es un conjunto de dispositivos electrónicos conectados entre sí a través de un medio, que intercambian información y comparten recursos.

Los protocolos de red definen un conjunto de reglas y convenciones para llevar a cabo la comunicación entre dispositivos en una red, incluyendo mecanismos para la autenticación, la transferencia de datos, la detección de errores y la corrección de errores, y la señalización.

Este concepto se presenta como una introducción a los conceptos básicos de redes, que abarca algunos de los protocolos más comunes, como TCP/IP, UDP y SSH, así como los tipos y topologías de redes, y ataques comunes a redes.

4.3.1. Motivación

El conocimiento de las redes ocupa un lugar destacado Entre las habilidades esenciales para llevar a cabo un pentesting efectivo, debido a su importancia fundamental en la seguridad de los sistemas.

Antes de realizar una prueba de penetración, es vital identificar los objetivos y las áreas de interés en la red, ya que sin esta información, sería difícil determinar qué atacar y dónde centrar los esfuerzos de prueba.

El conocimiento de las direcciones IP y los protocolos utilizados por los sistemas en la red permite a los profesionales de seguridad seleccionar las herramientas y técnicas adecuadas para las pruebas, porque diferentes sistemas y servicios pueden requerir enfoques de ataque específicos, y esta elección informada es esencial para simular con precisión los métodos utilizados por los atacantes reales.

Por otra parte, comprender la topología de la red ayuda a identificar puntos débiles, rutas alternativas y posibles vectores de ataque.

4.4. Análisis de tráfico

Esta es una técnica de ciberseguridad que implica la monitorización y el examen de los datos que fluyen a través de una red de comunicaciones. Puede incluir la recopilación de información sobre el origen y el destino de los datos, el tipo de protocolo utilizado, la cantidad de datos transmitidos y otros metadatos relacionados con el tráfico, etc.

Este concepto se presenta como una descripción del proceso de análisis de tráfico, que abarca la captura de tráfico de red, el análisis de paquetes y el uso de herramientas de análisis de tráfico, como Wireshark.

4.4.1. Motivación

El análisis de tráfico es una habilidad esencial para los profesionales de la seguridad informática, ya que permite identificar y comprender el tráfico de red, lo que facilita la detección de anomalías y de posibles amenazas.

Durante una prueba de penetración, puede ayudar a identificar sistemas y servicios en la red, así como a comprender cómo interactúan entre sí. Esto permite a los pentesters identificar posibles vulnerabilidades y vectores de ataque, y seleccionar las herramientas y técnicas adecuadas para la prueba.

4.4.2. Laboratorio

Este entorno permite al usuario usar Wireshark en el propio navegador, alojando el servicio en otro puerto del servidor, accesible desde la propia plataforma. Contiene 4 capturas de tráfico que involucran los protocolos ICMP, ARP, TCP y FTP.

La página de la plataforma hace un recorrido y explicación de 2 de ellas:

La captura `SSL.pcap` contiene el intercambio de datos de una conexión TCP, donde se produce el proceso de *handshake* de SSL/TLS, que permite el cifrado de la conexión. El usuario puede usar Wireshark para explorar distintos detalles sobre las versiones de TLS utilizadas y cómo se han decidido utilizar. Se muestra cómo se han obtenido los algoritmos de cifrados y las características de la conexión.

La captura `filezilla.pcap` contiene el intercambio de datos de una conexión FTP, donde se produce el proceso de autenticación del usuario, que permite el acceso al servidor FTP. El usuario puede usar Wireshark para explorar distintos detalles sobre el intercambio de datos, como el usuario y contraseña, así como el uso de comandos FTP para la gestión de archivos. También es posible observar el contenido de los ficheros enviados.

Las otras 3 capturas, llamadas `Internet.pcap`, `ping.pcap` y `SSH.pcap` muestran información sobre otros protocolos útiles como ICMP, SSH y ARP.

4.5. Introducción al OSINT

La inteligencia de código abierto (*Open Source INTelligence*) es una técnica de investigación que se basa en la obtención de información a partir de fuentes de acceso público.

Este concepto se presenta como una introducción a OSINT, abarcando la búsqueda de información en distintas herramientas, como los Google Dorks, Shodan.io, VirusTotal y WHOIS.

4.5.1. Motivación

Esta técnica se utiliza en el sector de la ciberseguridad para obtener información sobre un objetivo, como puede ser una persona, una empresa o una organización, con el objetivo de realizar un ataque posterior; pero también, sobre posibles intentos de Phishing, ingeniería social, etc. detectando elementos sospechosos como IPs o direcciones de correo electrónico.

4.5.2. Laboratorio

Este entorno presenta un proyecto de Python con el que extraer información de una IP de varias fuentes de información, entre las que se encuentran Shodan, VirusTotal y WHOIS.

El proyecto consiste en una interfaz de línea de comandos que permite al usuario introducir una IP o una lista de IPs (en un archivo de texto) y obtener información de cada una de ellas usando las fuentes mencionadas, entre otras.

Nota La información de Shodan.io y VirusTotal se obtiene a través de sus APIs públicas, por lo que es necesario tener una cuenta en cada una de ellas y obtener una clave de API para poder usarlas. No usar una clave no afectará al programa más allá de no usar esas 2 fuentes de información.

La herramienta desarrollada para este laboratorio se explica en mayor profundidad en el apéndice C.2.1.

4.6. Esteganografía

La esteganografía es una técnica de ocultación de información dentro de un archivo, como puede ser una imagen, un vídeo o un archivo de audio, lo que facilita la transmisión de datos confidenciales sin ser detectados.

Este laboratorio presenta una definición de esteganografía y muestra de herramientas comunes, como `steghide` y `exiftool`.

4.6.1. Motivación

Esta técnica se utiliza en el sector de la ciberseguridad para ocultar información sensible dentro de archivos que no levanten sospechas, por lo que durante una prueba de penetración, los conocimientos de esteganografía pueden ayudar a descubrir información oculta en archivos multimedia.

4.6.2. Laboratorio

Este entorno contiene una serie de imágenes con información oculta en ellas, donde el usuario podrá identificar la información oculta y usar las herramientas mencionadas para extraer dicha información.

Toda la información oculta está conectada, por lo que es posible llevar a cabo un proceso similar al *footprinting* para obtener una información específica.

4.7. Fuerza bruta

La fuerza bruta es una técnica en la que se intenta adivinar una contraseña o credencial de acceso probando diferentes combinaciones hasta encontrar la correcta. Generalmente, usando un diccionario de contraseñas como `rockyou.txt`.

Este concepto se presenta como una descripción de la técnica de fuerza bruta, abarcando el uso de herramientas para descifrar contraseñas.

Este apartado es especialmente corto dada la naturaleza básica del concepto, ya que se consideró que no era necesario profundizar en él por su conexión con otros más relevantes, como por ejemplo el hash-cracking (sección 4.8).

4.7.1. Motivación

La fuerza bruta es una técnica muy utilizada en el sector de la ciberseguridad para descifrar contraseñas, ya que es relativamente sencilla de implementar y puede ser muy efectiva si se usa correctamente.

Este término es la base de muchos otros conceptos y está presente en muchas otras técnicas de penetración, como listado de directorios, enumeración de usuarios, etc.; por otra parte, es importante saber identificar y aplicar los diccionarios correctos.

4.7.2. Laboratorio

Este entorno presenta un sistema Linux vulnerable, ya que la contraseña del usuario `root` es una contraseña común y pertenece al diccionario `rockyou.txt`.

4.8. Hash-cracking

El *hash-cracking* es una técnica de ataque que se utiliza para obtener la contraseña original a partir de un hash.

Este concepto se presensta como descripción de la técnica de *hash-cracking*, abarcando el uso de herramientas para descifrar información.

4.8.1. Motivación

Las contraseñas y otros datos sensibles, rara vez se almacenan en texto plano debido a los riesgos de seguridad asociados; en su lugar, se transforman en valores hash utilizando algoritmos de hash criptográficos.

Conocer técnicas de *hash-cracking* permite a los pentesters evaluar la calidad de las contraseñas utilizadas por los usuarios en un sistema. Si las contraseñas se descifran con relativa facilidad, esto indica que los usuarios están utilizando contraseñas débiles, lo que representa un riesgo de seguridad significativo.

Por último, si los pentesters logran descifrar contraseñas y acceder a cuentas o sistemas protegidos, pueden demostrar el impacto real que tendría un ataque exitoso. Esto es

especialmente valioso para concienciar a las partes interesadas sobre la importancia de mejorar la seguridad de las contraseñas y tomar medidas adecuadas.

4.8.2. Laboratorio

Este entorno presenta 3 bases de datos cuyas contraseñas han sido almacenadas usando 3 algoritmos de hash diferentes: MD5, SHA1 y SHA256.

El usuario puede emplear las herramientas descritas en la plataforma para identificar el tipo de hash usado en cada base de datos, y llevar a cabo una extracción de las contraseñas de cada una.

Además, se le proporciona al usuario una breve guía de comandos SQL para usar, ya que la finalidad de este laboratorio no es el manejo de bases de datos.

4.9. Criptografía

La criptografía es una técnica que se utiliza para cifrar y descifrar información, cuyo objetivo es que solo el emisor y el receptor puedan acceder a ella.

Este concepto se presenta como una introducción a la criptografía, abarcando algunos tipos de cifrado y su clasificación, así como el uso de herramientas de criptografía para cifrar y descifrar mensajes.

4.9.1. Motivación

La criptografía está presente en muchos aspectos de la seguridad informática, como la autenticación de usuarios y la firma digital, por lo que es importante conocer los conceptos básicos de la criptografía para comprender cómo funcionan estos mecanismos de seguridad.

4.10. Escalada de privilegios

La escalada de privilegios es una técnica de hacking que se utiliza para obtener acceso a sistemas o recursos que normalmente estarían restringidos por permisos de seguridad.

Este concepto se presenta como una descripción de la técnica de escalada de privilegios y posibles casos de uso de la misma.

4.10.1. Motivación

Esta técnica se utiliza cuando un atacante ya ha obtenido acceso a un sistema con un conjunto limitado de permisos, pero necesita obtener permisos adicionales para llevar a cabo acciones malintencionadas, resultando en una habilidad muy útil para la ejecución de pruebas de penetración, ya que si es posible obtener permisos de administrador o de otro usuario privilegiado, el atacante puede tener acceso a información confidencial, instalar malware o tomar el control total del sistema.

4.10.2. Laboratorio

Este entorno presenta una vulnerabilidad de permisos débiles; en esta ocasión, el fichero `/etc/shadow` tiene permisos de lectura para todos los usuarios, por lo que cualquier usuario puede leer el contenido del fichero, que contiene las contraseñas cifradas de los usuarios del sistema.

Una vez se lee el fichero, se puede observar que existen 2 usuarios: `root` y `user`, siendo este último el que maneja la conexión SSH para el usuario de la plataforma.

Además, haciendo un estudio del formato de los hashes almacenados, es posible llegar a la conclusión de que las contraseñas están usando MD5 crypt, por lo que es posible descifrarlas usando una herramienta de *hash-cracking*.

Obteniendo así, la contraseña del usuario `root` y pudiendo acceder al sistema como administrador.

4.11. Bypass

Un bypass es una vulnerabilidad que permite a un atacante evadir un mecanismo de seguridad, como la autenticación, para obtener acceso no autorizado a un sistema o recurso.

Este concepto se presensta como una descripción del término de bypass, a través de la vulnerabilidad CVE-2017-8386 [25], que permite a un atacante remoto omitir la autenticación en un servidor Git.

4.11.1. Motivación

Una vulnerabilidad de bypass puede permitir a un atacante evadir un mecanismo de seguridad, como la autenticación, para obtener acceso no autorizado a un sistema o recurso.

Esta técnica se utiliza en el sector de la ciberseguridad para obtener acceso a sistemas o recursos que normalmente estarían restringidos por permisos de seguridad, por lo que es importante conocer los conceptos básicos de los bypasses para comprender cómo funcionan estos mecanismos de seguridad y cómo se pueden evitar.

4.11.2. Laboratorio

Este laboratorio simula un servidor de Git con la vulnerabilidad CVE-2017-8386. Se requiere una clave SSH privada proporcionada en la plataforma para que se realice la conexión correctamente.

Una vez se hayan completado esos preparativos, se podrá llevar a cabo la explotación mencionada anteriormente y el usuario podrá explorar las posibilidades que ofrece dicho bypass.

4.12. Ransomware

Este es un laboratorio especial -algo más alejado del pentesting- que trata la peligrosidad de un ransomware, al mismo tiempo que se observa la sencillez con la que este puede causar mucho daño en un sistema.

4.12.1. Motivación

Se ha considerado importante usar una parte de este trabajo de fin de grado para concienciar a los usuarios sobre la amenaza real que supone el ransomware, ya que es una de las amenazas más comunes en la actualidad y puede causar mucho daño en un sistema.

4.12.2. Laboratorio

Este entorno contiene una muestra de ransomware muy simple, pero igualmente peligrosa si no se trata con cuidado. El ransomware se llama `stockholm.py` y ha sido desarrollado en Python.

El script se encuentra en el directorio del usuario `root` y realizará las siguientes acciones:

1. Creará una clave de cifrado aleatoria.
2. Buscará una carpeta llamada `infection/` en la carpeta del usuario `root`.
3. Usando la clave, cifrará todos los archivos de la carpeta `infection/` cuya extensión también fuera afectada por *WannaCry*.
4. Guardará la clave de cifrado de forma segura en un archivo llamado `clave.key`.

No obstante, como solo es una prueba de concepto, el script también puede devolver los archivos a su estado original utilizando la clave de cifrado proporcionada por el mismo (paso 1): ejecutar `stockholm.py -r {clave}` revertirá el cifrado.

El objetivo de este laboratorio es que el usuario observe el funcionamiento de un ransomware y expermiente con él para que pueda entender mejor cómo funcionan y cómo se pueden evitar.

CAPÍTULO 5

Resultados

5.1. Cumplimiento de los requisitos

Se muestran a continuación los requisitos funcionales definidos en el capítulo 3.1 y su cumplimiento en la plataforma.

Para ello se ha usado Gherkin [26], un DSL (*Domain-Specific Language*) que permite definir requisitos funcionales en un lenguaje natural, y que además, permite definir escenarios de prueba para comprobar el cumplimiento de dichos requisitos.

RF - 01	**Registro de usuarios**
El sistema debe permitir que los usuarios se registren y creen una cuenta en la plataforma para poder usar los laboratorios de pruebas.	
escenario	**Registro correcto de un usuario.**
cuando	Estoy en la página de registro.
entonces	Relleno el formulario de registro con datos válidos.
y	Pulso el botón de registro.
entonces	Un mensaje me indica que mi usuario ha sido creado correctamente.
cuando	Estoy en la página de inicio de sesión.
entonces	Puedo iniciar sesión con el usuario que acabo de crear.
escenario	**Registro incorrecto de un usuario.**
cuando	Estoy en la página de registro.
entonces	Relleno el formulario de registro con datos inválidos.
y	Pulso el botón de registro.
entonces	Un mensaje me indica que ha ocurrido un error al crear mi usuario.

RF - 02	**Inicio de sesión**
El sistema debe permitir que los usuarios inicien sesión en la plataforma para poder usar los laboratorios de pruebas.	
escenario	**Inicio de sesión correcto de un usuario.**
cuando	Estoy en la página de inicio de sesión.
entonces	Relleno el formulario de inicio de sesión con datos válidos.
y	Pulso el botón de inicio de sesión.
entonces	Soy redirigido a la página de inicio.
escenario	**Inicio de sesión incorrecto de un usuario.**
cuando	Estoy en la página de inicio de sesión.
entonces	Relleno el formulario de inicio de sesión con datos inválidos.
y	Pulso el botón de inicio de sesión.
entonces	Un mensaje me indica que ha ocurrido un error al iniciar sesión.

RF - 03	**Validación de los datos de registro**
El sistema debe validar los datos de registro de los usuarios para poder crear una cuenta en la plataforma.	
escenario	**Validación correcta de los datos de registro.**
cuando	Estoy en la página de registro.
entonces	Relleno el formulario de registro con datos válidos.
y	Pulso el botón de registro.
entonces	Un mensaje me indica que mi usuario ha sido creado correctamente.
escenario	**Validación incorrecta de los datos de registro.**
cuando	Estoy en la página de registro.
entonces	Relleno el formulario de registro con datos inválidos.
y	Pulso el botón de registro.
entonces	Un mensaje me indica que ha ocurrido un error al crear mi usuario.

RF - 04	**Consulta de documentación**
El sistema debe permitir que los usuarios consulten la documentación de la plataforma para poder aprender sobre los conceptos de ciberseguridad.	
escenario	**Consulta de documentación.**
cuando	Estoy en la página de laboratorios.
entonces	Puedo ver una lista de conceptos de ciberseguridad.
cuando	Estoy en la página de un concepto.
entonces	Puedo ver una descripción del concepto asociado.
y	Avanzando en la página.
entonces	Puedo ver un laboratorio asociado (si lo hubiera).

RF - 05	**Inicio de laboratorios**
El sistema debe permitir que los usuarios inicien y detengan laboratorios de pruebas para poder practicar los conceptos de ciberseguridad.	
escenario	**Inicio correcto de un laboratorio de pruebas.**
cuando	Estoy en la página de un concepto.
y	Existe un laboratorio de pruebas asociado.
y	He iniciado sesión en la plataforma.
y	No tengo ningún laboratorio de pruebas activo.
y	Pulso el botón de inicio del laboratorio.
entonces	Se crea un laboratorio de pruebas alojado en un puerto.
escenario	**Inicio incorrecto de un laboratorio de pruebas**
cuando	Estoy en la página de un concepto.
y	Existe un laboratorio de pruebas asociado.
y	No he iniciado sesión en la plataforma.
entonces	Un mensaje me indica que debo iniciar sesión.
cuando	Estoy en la página de un concepto.
y	Existe un laboratorio de pruebas asociado.
y	He iniciado sesión en la plataforma.
y	Tengo un laboratorio de pruebas activo.
entonces	Un mensaje me indica que ya tengo un laboratorio activo.

RF - 06	**Tiempo de vida de un laboratorio**
El sistema debe permitir que los usuarios inicien y detengan laboratorios de pruebas para poder practicar los conceptos de ciberseguridad.	
escenario	**Tiempo de vida alcanzado (1 hora).**
cuando	Estoy dentro del laboratorio.
y	El laboratorio alcanza su tiempo de vida.
entonces	El laboratorio se detiene automáticamente.
y	Soy expulsado de la sesión SSH.

RF - 07	**Número máximo de laboratorios**
El sistema debe permitir que los usuarios inicien y detengan laboratorios de pruebas para poder practicar los conceptos de ciberseguridad.	
escenario	**Número máximo de laboratorios alcanzado.**
cuando	Estoy en la página de un concepto.
y	Existe un laboratorio de pruebas asociado.
y	He iniciado sesión en la plataforma.
y	Tengo un laboratorio de pruebas activo.
entonces	Un mensaje me indica que ya tengo un laboratorio activo.

RF - 08	**Uso de un laboratorio iniciado**
El sistema debe permitir que los usuarios inicien y detengan laboratorios de pruebas para poder practicar los conceptos de ciberseguridad.	
escenario	**Quiero usar un laboratorio que aún no he detenido.**
cuando	Estoy en la página del concepto del laboratorio.
y	Existe un laboratorio de pruebas asociado.
y	He iniciado sesión en la plataforma.
y	Tengo el laboratorio de pruebas activo.
y	El laboratorio no ha alcanzado su tiempo de vida.
entonces	Un mensaje me indica que el laboratorio está activo.
y	Puedo conectarme usando SSH.

RF - 09	**Detener de laboratorios**
El sistema debe permitir que los usuarios detengan laboratorios de pruebas para poder practicar los conceptos de ciberseguridad.	
escenario	**Detención correcta de un laboratorio de pruebas.**
cuando	Estoy en la página de un concepto.
y	Existe un laboratorio de pruebas asociado.
y	He iniciado sesión en la plataforma.
y	Tengo el laboratorio de pruebas activo.
y	Pulso el botón de detención del laboratorio.
entonces	Se detiene el laboratorio de pruebas alojado en un puerto.
escenario	**Detención incorrecta de un laboratorio de pruebas**
cuando	Estoy en la página de un concepto.
y	Existe un laboratorio de pruebas asociado.
y	No he iniciado sesión en la plataforma.
entonces	Un mensaje me indica que debo iniciar sesión.

5.2. Conclusiones

La realización de este proyecto me ha permitido profundizar en el mundo de la ciberseguridad, así como en el desarrollo web y la virtualización con Docker. Considero que me ha servido para ampliar mis conocimientos en estas áreas y para aprender nuevas técnicas y herramientas que no conocía.

Personalmente, tenía 2 objetivos principales al llevar a cabo este proyecto: ampliar mi conocimiento sobre ciberseguridad y pentesting; y aprender sobre Docker.

Respecto al primer objetivo, he podido adquirir más conocimientos sobre algunos conceptos que ya conocía, como: OSINT, esteganografía y fuerza bruta; pero también he podido avanzar en aquellos otros conceptos en los que considero que era necesario aprender más, como: el análisis de tráfico, la escalada de privilegios y el hash-cracking.

Por otra parte, también me ha resultado útil para revisar algunos conceptos tratados durante la carrera o debido a la misma, como Bash scripting y redes.

Respecto al segundo objetivo, he aprendido a usar Docker y he sido testigo no solo de la potencia de esta herramienta, sino de la infinidad de posibilidades que esta plantea, tanto en el ámbito de desarrollo, como en el de producción de sistemas y servicios.

Si bien no ha sido necesario un uso avanzado de Docker para este proyecto, sí he investigado y aprendido acerca de todas sus otras funcionalidades, como los volúmenes, las redes y los ficheros `dockercompose.yml`.

Por último, con este proyecto también he adquirido experiencia en el desarrollo web a través de una de las herramientas más usadas: WordPress. Nunca trabajé con ella, por lo que considero que ha sido una buena oportunidad para aprender más sobre esta herramienta y cómo funcionan los sitios web construidos sobre ella.

5.3. Futuras líneas de trabajo

El desarrollo de este proyecto ha dado lugar a muchas ideas y posibles mejoras que no se han podido aplicar por falta de tiempo o por falta de conocimientos. A continuación se muestra una lista con las ideas más relevantes e influyentes que podrían ser aplicadas en un futuro:

5.3.1. Traslado del proyecto a un servicio de alojamiento

Todo el proyecto está alojado en un servidor local, por lo que el siguiente paso natural sería trasladar la plataforma a un servicio de alojamiento, como por ejemplo alguno de los mencionados en la sección 2.3.3, especialmente Google Compute Engine al considerarse el más compatible con la naturaleza y estructura del proyecto.

Esto permitiría que la plataforma estuviera disponible públicamente para cualquier usuario con acceso a Internet, facilitando aún más su consumo y su uso.

5.3.2. Expansión del contenido de pentesting

Actualmente la plataforma describe 12 conceptos que cubren los aspectos descritos en la sección 4, pero se podrían describir y añadir más conceptos junto a sus laboratorios, para ampliar el contenido de la plataforma y ofrecer más experiencias instructivas a los usuarios.

Por ejemplo: simulación de nuevos ataques, explotación de vulnerabilidades, etc.

5.3.3. Optimización de la plataforma

La plataforma está construida sobre WordPress, pero se podría plantear la posibilidad de desarrollarla desde cero, ya que WordPress es una herramienta muy potente, pero también muy pesada, por lo que podría ser interesante desarrollar una plataforma más ligera y personalizada.

Por otra parte, también existen herramientas alternativas que han sido diseñadas específicamente para crear plataformas de CTFs, como: Mellivora [27], CTFd [28], Root The Box [29], etc.

Importante Cabe destacar que **este proyecto no es una plataforma de CTFs**, ni debe serlo, ya que los laboratorios planteados quedan a disposición completa del usuario para que los use como desee. Sin embargo, sí podría ser interesante investigar estas herramientas y quizás, aplicar modificaciones para que las pruebas virtualizadas no requieran de una bandera, convirtiendo la plataforma en una herramienta de aprendizaje y no de competición.

5.3.4. Generalización del contenido

El contenido inicial de la plataforma está enfocado al pentesting, pero podría establecerse una temática genérica para abarcar muchos más conceptos de distintas áreas de la seguridad.

Por ejemplo: administración de sistemas, análisis de malware, desarrollo seguro, hacking web, etc. diferentes ramas y especializaciones de ciberseguridad.

Esto daría lugar a una plataforma más eficiente y parecida a otras ya existentes, pero manteniendo la diferencia que se fomentaba al inicio del proyecto en la sección 1.2: una plataforma ligera, *open-source* y fácilmente extensible, que permita a los usuarios aprender y practicar conceptos de ciberseguridad mediante laboratorios.

5.3.5. Organización del contenido

Elaboración de rutas o módulos de aprendizaje que permitan al usuario seguir un orden lógico al adquirir conocimientos, en lugar de elegir un conceptos al azar o tratar de asumir un orden por su cuenta.

Por ejemplo: un módulo de introducción a redes, otro de ciber-defensa, etc; una ruta para aprender sobre herramientas de fuerza bruta y explorar sus posibilidades, otra para herramientas como IDS o IPS, etc.

Esto podría favorecer a la velocidad de aprendizaje del usuario, ya que no tendría que pensar en qué concepto aprender a continuación, sino que podría seguir una ruta de aprendizaje establecida por la plataforma, diseñada para que las conexiones entre las temáticas sean más claras y lógicas.

Bibliografía

1. *GitHub*, (www.github.com).
2. *GitHub Pages*, (pages.github.com).
3. *Vercel*, (www.vercel.com).
4. *OVHcloud*, (www.ovhcloud.com).
5. *Heroku*, (www.heroku.com).
6. *Vagrant Cloud*, (www.app.vagrantup.com/boxes/search).
7. *Docker Hub*, (hub.docker.com).
8. *WordPress*, (www.wordpress.com).
9. *TIME*, (www.time.com).
10. *BCC America*, (www.bbcamerica.com).
11. *Divi*, (www.elegantthemes.com/gallery/divi).
12. *Hack The Box*, (www.hackthebox.eu).
13. *HTB Academy*, (www.academy.hackthebox.com).
14. *TryHackMe*, (www.tryhackme.com).
15. *VulnHub*, (www.vulnhub.com).
16. *OverTheWire*, (www.overthewire.org/wargames).
17. *Academia Hacker de Incibe - Retos descargables*, (www.incibe.es/academiahacker/retosdescargables).
18. S. Dash, F. Massacci, A. Sabetta y D. R. Dos Santos, *Cyber Security Testbeds and Malware Testing*, 2021, (www.securitylab.disi.unitn.it/doku.php?id=malware_analysis).
19. *Alpine Linux*, (www.alpinelinux.org).
20. *Debian*, (www.debian.org).
21. *Astro*, (www.astro.build).
22. *Drupal*, (www.drupal.org).
23. *SQLite*, (www.sqlite.org/index.html).
24. *MySQL*, (www.mysql.com).
25. *CVE-2017-8386*, (cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-8386).
26. *Gherkin*, (www.cucumber.io/docs/gherkin/reference).

27. *Mellivora*, (www.github.com/Nakiami/mellivora).
28. *CTFd*, (www.ctfd.io).
29. *Root The Box*, (www.github.com/moloch--/RootTheBox).
30. U. Gascón González, *Docker Seguro*.
31. D. Santo Orcero, *La Biblia en LaTeX*.
32. *El Gobierno andaluz creará el Centro de Ciberseguridad de Andalucía con una inversión de más de 60 millones de euros*, (www.juntadeandalucia.es/presidencia/portavoz/161360).
33. *Project Jupyter*, (www.jupyter.org).
34. *JupyterHub*, (www.jupyter.org/hub).
35. *Docker Compose*, (www.docs.docker.com/compose).
36. *GenerateWP*, (www.generatewp.com/post-type).
37. *Vagrant*, (www.vagrantup.com).
38. *WPvivid Backup Plugin*, (www.es.wordpress.org/plugins/wpvivid-backuprestore).
39. *Draw.io*, (www.draw.io).
40. *Shots.so*, (www.shots.so).

Apéndices

APÉNDICE A

Toma de decisiones

Esta sección tratará distintas problemáticas que se han abordado durante el desarrollo del proyecto, de forma que se puedan entender mejor las decisiones tomadas y redactadas en los capítulos anteriores de esta memoria.

No se incluyen aquellos problemas que se hayan resuelto con investigación, como por ejemplo: saber usar algunas funcionalidades de WordPress, qué casos incluir en los laboratorios, etc.

A.1. Gestión local de contenedores Docker

Durante todo el documento se ha hablado de la funcionalidad del proyecto: una plataforma web que permite a los usuarios acceder a laboratorios virtuales donde poner en práctica los conocimientos adquiridos en la propia plataforma, siendo dichos laboratorios contenedores Docker que el usuario puede iniciar o detener a voluntad desde el sitio web, aunque con ciertas limitaciones.

Sin embargo, esto planteaba un problema: cómo se podrían gestionar esos contenedores Docker.

A.1.1. Conexión entre el sitio web y Docker

El objetivo se trataba de algo tan simple como levantar y destruir contenedores Docker, sin embargo, esto debía hacerse de forma dinámica, ya que el resultado final esperado era tener un sistema donde los contenedores no estuvieran permanentemente activos, sino que fueran activados y desactivados en función de su uso.

El uso de estrategias como Kubernetes excedían a las propias necesidades del proyecto, ya que al tratarse de una prueba de concepto no se requería gestionar un gran número de contenedores, ni tampoco era necesario que se organizaran en clústeres; por tanto, se decidió que los contenedores se gestionarían de forma local.

Tampoco se usaron APIs ni librerías, ya que se consideró que no era necesario, y que se podía hacer uso de los propios comandos del sistema para gestionar los contenedo-

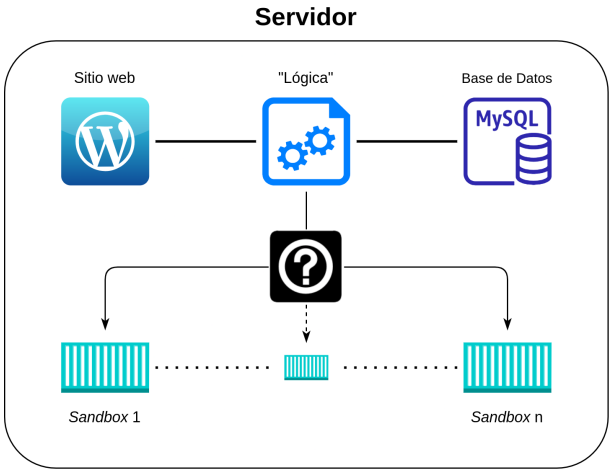


Figura A.1: Diagrama sobre el problema de conexión entre el sitio y los contenedores

res Docker; de nuevo, porque la idea en sí resultaba muy simple, aunque más tarde su complejidad fue creciendo.

El resultado final fue el uso del siguiente comando del sistema, que se ejecuta usando `exec()` en el fichero `functions.php` del sitio web:

```
exec("docker run --rm -p $puerto/22 -d $laboratorio:latest");
```

El comando recibe un puerto libre del sistema y el nombre del laboratorio (imagen Docker) para crear el contenedor, indicado como variables PHP en la instrucción anterior. La opción `--rm` se usa para que el contenedor se destruya automáticamente al detenerse, y la opción `-d` para que se ejecute en segundo plano (aún en segundo plano devolverá la ID del contenedor creado).

A.1.2. Mapeo de puertos sin solapamientos

Por otro lado, se presentaba el siguiente problema, ¿cómo se podría gestionar el mapeo de puertos de los contenedores Docker de forma automática?

Al no usar Kubernetes, no se podía usar el mapeo de puertos de forma dinámica, ya que los contenedores se gestionaban de forma local y no se podía acceder a ellos desde fuera de la máquina donde se ejecutaban, por lo que se decidió que los contenedores se gestionarían de forma local.

La opción `-p` de Docker permite mapear puertos de forma estática, lo que implica que si se usaba un puerto que ya estuviera en uso, la creación del contenedor fallaría.

Por otra parte, es posible usar un rango de puertos con esa opción, lo que permite a Docker mapear un puerto del contenedor a cualquier puerto libre en el rango específicado de forma automática y aleatoria (la selección no es secuencial); esta parecería la opción perfecta pero tenía una gran limitación: los puertos libres de un rango de puertos solo son conocidos entre los contenedores de la misma imagen, y por tanto, como cada laboratorio es una imagen Docker de la que se construyen contenedores para los usuarios, esto quiere decir que no era posible determinar todos los puertos libres para todas las imágenes.

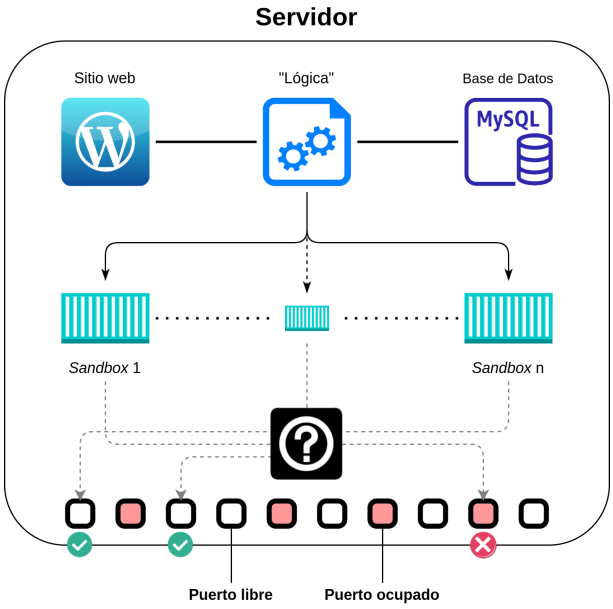


Figura A.2: Diagrama sobre el problema de mapeo de puertos

Otra opción que se contempló fue la de usar el mismo rango para todas las imágenes, pero uno tan amplio que se redujera todo lo posible la posibilidad de colisiones de puertos entre contenedores de distintas imágenes; sin embargo, esta opción fue descartada porque, aunque pudiera haber sido efectiva en esta prueba de concepto, no dejaba de existir la posibilidad de que se produjeran colisiones, y por tanto, no se consideró una solución aceptable.

El uso de ficheros `docker-compose.yml` también era muy restrictivo, ya que no permitía la gestión dinámica de puertos; para ser más específicos, este fichero almacena una configuración a la que se puede llegar usando combinaciones de opciones y valores de comandos Docker, por lo que no añade ninguna funcionalidad adicional, sino que solo permite almacenar una configuración. Aquí se reptite exactamente el mismo problema de las colisiones descrito en el párrafo anterior.

Finalmente, se optó por generar puertos disponibles usando código en `functions.php`: se define un rango inicial de puertos y para cada laboratorio que quiere iniciar un usuario, se busca en la base de datos qué puertos están siendo ocupados por otros laboratorios, se calcula la diferencia entre el rango inicial y los puertos ocupados, y se selecciona un puerto aleatorio dentro de ese rango.

Como el puerto obtenido se usa para el laboratorio, al almacenarse esa información en la base de datos, el puerto queda registrado; del mismo modo cuando el usuario o la tarea de cron destruyan un laboratorio, los datos serán liberados de la base de datos y el puerto volverá a estar disponible para su uso.

A.2. Limitaciones del uso de contenedores y Docker

Cuando una persona piensa en la ciberseguridad, se imagina la clásica imagen de un hacker tecleando comandos en una pantalla llena de terminales, pero lo cierto es que

aunque muchas herramientas se usan como scripts o comandos, no todas son así y algunas requieren una interfaz gráfica para su uso.

Sin embargo, ocurre que para este caso concreto, no podrían usarse contenedores para aplicaciones que se usen con GUI. Los contenedores pueden llegar a ejecutar aplicaciones gráficas, pero esto implicaría que los usuarios descargaran la imagen y ellos crearan los contenedores, porque si no, estarían abriendo la aplicación en el servidor, donde no tienen acceso más allá de del acceso al contenedor por SSH.

En algunas ocasiones es posible ejecutar una GUI a través del navegador, lo que sumado al mapeo de puertos, podría permitir que el usuario accediera a la aplicación, pero esto no es posible en todos los casos y además, no es una solución que se pueda aplicar de forma genérica.

Por otra parte, existe la limitación actual de solo abrir y mapear un puerto por contenedor, teniendo que tomar la decisión sobre qué hacer en cada caso o qué finalidad tiene el laboratorio: entorno virtualizado con herramientas preinstaladas y casos de uso para dichas herramientas, o contenedor vulnerable con el que trabajar desde una máquina virtual propia del usuario.

A.3. Procesos bajo el usuario `www-data`

Al no usar una herramienta externa como Kubernetes, todos los procesos se realizan de forma local, por lo que el usuario que ejecuta los comandos es el usuario que ejecuta el servidor web, en este caso, `www-data`.

Esto quiere decir que para que el proyecto funcione, el usuario `www-data` debe ser el que inicie el servicio de Docker, por lo que debe tener permisos para ello.

```
$ sudo -u www-data systemctl start docker
```

```
sudo: ejecutar un comando como otro usuario.
    -u: especificar el usuario.
systemctl: gestionar servicios del sistema.
```

Por otra parte, también necesario añadir el usuario `www-data` al grupo `docker` para que pueda ejecutar los comandos de Docker.

```
$ sudo usermod -aG docker www-data
```

```
sudo: ejecutar un comando como otro usuario.
usermod: modificar un usuario.
    -aG: añadir (a) a un grupo (G).
```

Una vez añadido, puede comprobarse que el usuario `www-data` pertenece al grupo `docker` usando el comando `groups` o el comando `id`.

```
$ groups www-data

www-data : www-data docker
```

```
groups: mostrar los grupos a los que pertenece un usuario.

Este comando solo muestra los grupos a los que pertenece el
   usuario indicado.
```

```
$ id www-data

uid=33(www-data) gid=33(www-data) groups=33(www-data),999(docker)
```

```
id: mostrar información sobre un usuario.

Este comando muestra diversas características del usuario:
    uid: identificador de usuario en el sistema.
    gid: identificador de grupo de usuario en el sistema.
    groups: identificadores de grupos del usuario.
```

Además, del mismo modo, tendrá que crear previamente todas las imágenes de los laboratorios para que aparezcan disponibles en Docker y pueda utilizarlas: las imágenes Docker creadas por un usuario solo son visibles para ese usuario, por lo que si el usuario `www-data` no crea las imágenes, no podrá usarlas.

Durante el desarrollo del proyecto se creó un repositorio para la creación de los laboratorios y se incluyó un script con el que actualizar las imágenes de laboratorios en el servidor, así como añadir laboratorios que no estuvieran previamente. Este script automatiza el proceso de creación de imágenes, por lo que puede ejecutarse del mismo modo que se ha mostrado con las capturas anteriores.

```
$ sudo -u www-data ./instalar.sh

Imagen: 01_intro-linux
    Duplicada.
    Actualizando...
    Actualizada.

Imagen: 02_intro-bash
    Duplicada.
    Actualizando...
    Actualizada.

Imagen: 03_intro-redes
    Creando...
    Creada.

Imagen: (...)

Laboratorios creados:
```

```
    * 01_intro-linux
    * 02_intro-bash
    * 03_intro-redes
    * (...)
```

Y por último, pasa lo mismo con la tarea de cron que automatiza la destrucción de los laboratorios: debe ser el usuario `www-data` quien defina y active la tarea de cron. La tarea de cron definida para este proyecto puede encontrarse en la sección 3.2.

```
    $ sudo -u www-data crontab -e
```

```
    sudo: ejecutar un comando como otro usuario.
        -u: especificar el usuario.

    crontab: gestionar las tareas de cron de un usuario.
        -e: editar las tareas de cron del usuario.
```

APÉNDICE B

Manual de usuario

B.1. Plataforma

Esta sección depende en su mayoría de la experiencia del usuario con WordPress; no obstante, se incluyen instrucciones para las funcionalidades más importantes y relacionadas con la elaboración del proyecto.

Se debe asumir que todo lo establecido en esta sección hace referencia sola y exclusivamente a la plataforma web y su contenido.

B.1.1. Anotaciones previas

Copia de seguridad con WPvivid

WPvivid [38] es un plugin de WordPress que permite realizar copias de seguridad de un sitio web, así como restaurarlas.

WordPress funciona almacenando toda su configuración y contenido en una base de datos, por lo que el plugin se encarga, ensencialmente, de hacer una copia de la base de datos y de los ficheros de la plataforma en formato `.zip`.

Existe un fichero de copia de seguridad de la plataforma hecha con WPvivid, llamado `html-wpvivid.zip`, que se puede usar tanto para restaurar los datos de la plataforma, como para duplicarla en otro sitio web de WordPress; para esta segunda opción, se recomienda seguir los siguientes pasos:

1. Crear un nuevo sitio web de WordPress.
2. Instalar el plugin WPvivid.
3. Restaurar la copia de seguridad en el nuevo sitio web.

Esto convertirá cualquier sitio web de WordPress vacío en una copia exacta del sitio web de la plataforma actual.

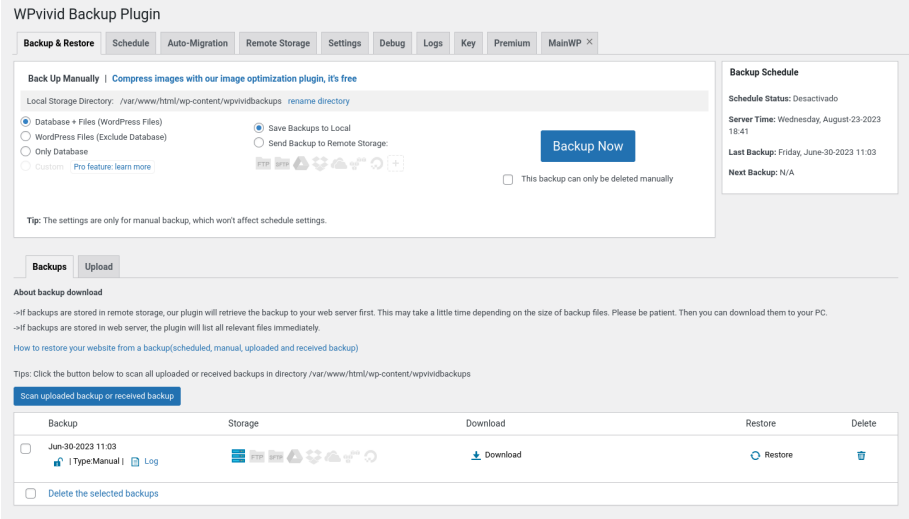


Figura B.1: Gestión de copias de seguridad con WPvivid

Importante Cabe destacar que, como se mencionó anteriormente, se copiará toda una base de datos, sobreescribiendo toda la información del sitio original, incluyendo las credenciales de los usuarios y administradores; por ello, la entrega también incluye una copia de las credenciales de un usuario administrador que se aconseja encarecidamente modificar una vez se haya importado la copia de seguridad.

B.1.2. Añadir un laboratorio

WordPress permite la creación de contenido personalizado, más allá de sus tipos predefinidos (Páginas, Entradas, etc.), por ello se ha creado un tipo de contenido personalizado llamado *Laboratorios*, disponible desde el menú lateral.

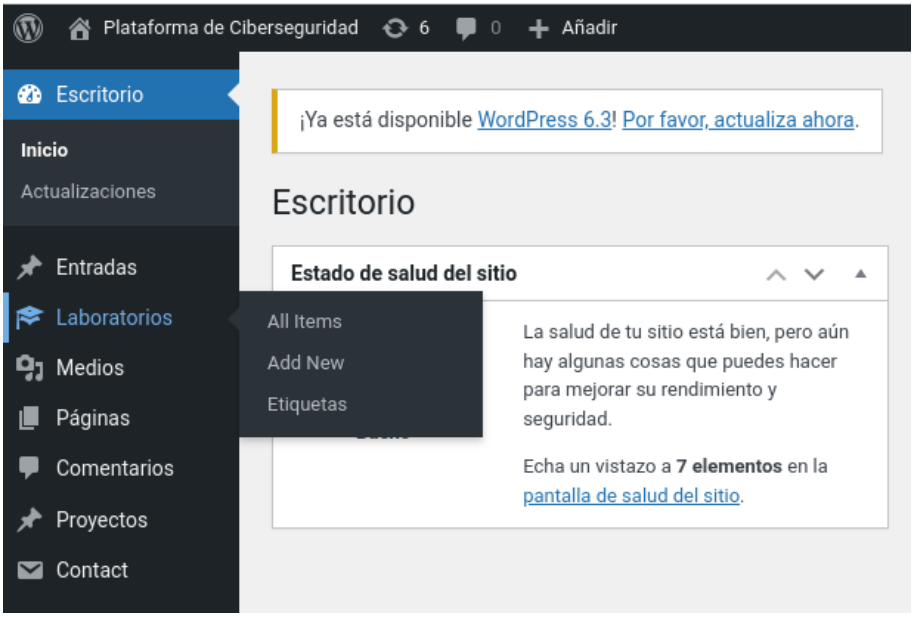


Figura B.2: Menú de los laboratorios

Una vez dentro, se puede crear un nuevo laboratorio usando el botón *Añadir nuevo* en la parte superior o usando el desplegable anterior.

Un laboratorio solo consta de 2 elementos: un texto descriptivo del concepto que se está tratando, que puede estar escrito en Markdown o HTML (recomendable este último); un identificador de un laboratorio, en la parte inferior, que no es más que **el nombre de la imagen Docker del sistema** que corresponde al laboratorio.

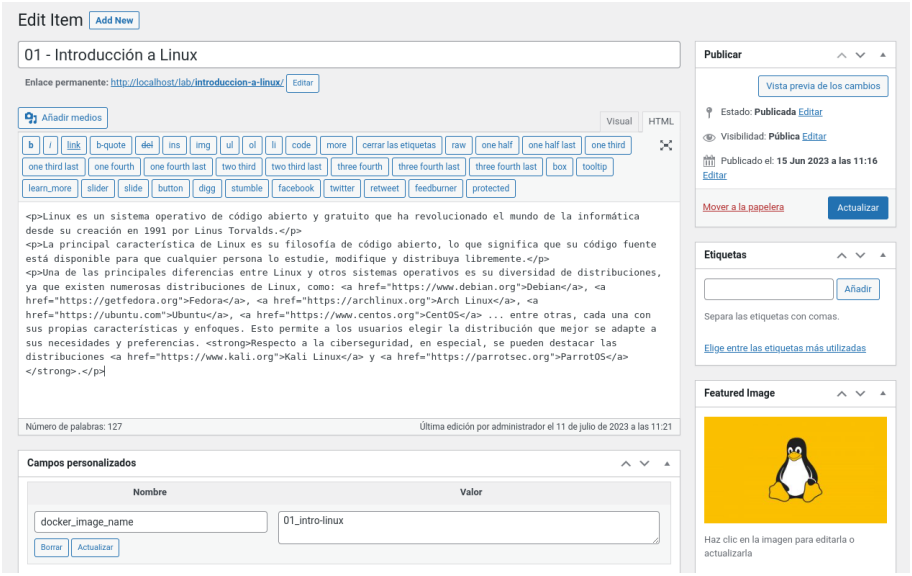


Figura B.3: Creación de un laboratorio

También es posible añadir opcionalmente una imagen de cabecera que se mostrará en la lista de laboratorios disponibles. Esta imagen debe ser local y una vez publicada en el sitio, formará parte de los archivos locales del mismo.

Nota Consulte la sección B.2.3 para saber más sobre la gestión de los laboratorios.

B.1.3. Modificar o eliminar un laboratorio

Cualquier laboratorio se puede modificar desde la lista de laboratorios disponibles, accediendo a la opción *Laboratorios* del menú lateral.

Modificar o eliminar un laboratorio consiste exclusivamente en el uso de las opciones de edición de contenido de WordPress, por lo que no se requieren instrucciones adicionales.

Sin embargo, se debe tener en cuenta que esta modificación solo afecta al contenido de la plataforma, por lo que si se desea aplicar alguna modificación al laboratorio en sí, será necesario modificar la imagen Docker correspondiente.

Nota Si se modifica el nombre de la imagen Docker o esta se elimina, recuerde que también debe modificar el atributo `docker_image_name` del *Laboratorio* de WordPress.

B.2. Laboratorios

Los laboratorios se han llevado a cabo en un repositorio distinto y será necesario un *clon* del mismo en el sistema para que la plataforma pueda referenciarlos.

La conexión que se establece entre la plataforma y los laboratorios virtualizados es la siguiente: **la plataforma solo se encarga de crear y destruir contenedores Docker, por lo que los laboratorios son imágenes Docker en el servidor donde se aloja la plataforma**.

Esto quiere decir que será necesario previamente construir las imágenes de los laboratorios en el servidor, para que la plataforma pueda usarlas.

Nota Como es la plataforma la que construye y destruye los laboratorios a petición de sus usuarios, todos los comandos de Docker de esta sección se deben ejecutar como el usuario `www-data`, ya que es el usuario que ejecuta el servidor web y por tanto, el que ejecuta los comandos de Docker.

B.2.1. Construir los laboratorios originales

Si se quiere usar los laboratorios originales, la forma más eficaz es ejecutar el script de instalación como se describe en la sección B.2.3.

B.2.2. Definir un laboratorio nuevo

Se entiende como *laboratorio* a una carpeta dentro de `labs/` cuyo contenido es, como mínimo, los siguientes ficheros:

- `Dockerfile`: instrucciones para crear la imagen Docker del laboratorio.
- `README.html`: descripción del laboratorio en formato HTML.

Por tanto, crear un laboratorio es tan sencillo como crear un fichero `README.html` que explique el concepto a tratar (teoría) y un fichero `Dockerfile` que contenga las instrucciones para crear la imagen Docker del que extraer posteriormente los contenedores (práctica).

Adicionalmente, algunos laboratorios también cuentan con una carpeta `resources/` que contiene recursos necesarios para la construcción de la imagen Docker.

Nota Cabe destacar que todos los ficheros descriptivos `README.html` han sido generados a partir de un fichero más simple: `README.md`, escrito en Markdown. Este archivo, pese a no ser necesario, está incluido en todos los laboratorios del repositorio; esto se debe a que el repositorio está publicado en GitHub y al llamarse estrictamente *README.md*, su contenido se procesa y se muestra en la web, lo que permite una mejor visualización del contenido del laboratorio.

B.2.3. Construir un laboratorio nuevo

Una vez se ha definido un laboratorio, existen 2 alternativas para construir el laboratorio:

Docker

Usando el comando de Docker para construir imágenes Docker:

```
$ sudo -u www-data docker build {ruta} -t {nombre}
```

```
sudo: ejecutar un comando como otro usuario.
    -u: especificar el usuario.
docker: comando para gestionar Docker.
    build: subcomando para construir una imagen Docker.
        -t: etiqueta (nombre) para la imagen Docker.
```

Nota El nombre de la imagen Docker será la que se use en el atributo `docker_image_name` del *Laboratorio* de WordPress.

Script de instalación

El repositorio incluye un script de instalación que permite tanto construir por primera vez nuevos laboratorios, como actualizar aquellos que hayan sido modificados.

Este script es `/labs/instalar.sh` y se debe ejecutar desde su ubicación:

```
$ cd labs/
$ chmod +x instalar.sh
$ ./instalar.sh
```

```
cd: cambiar de directorio.
chmod: modificar los permisos de un fichero.
    +x: añadir permisos de ejecucion.

instalar.sh: crea o actualiza cada laboratorio de labs/.
    Si existe una imagen con el mismo nombre, se actualiza;
    en caso contrario, se crea una nueva imagen Docker.
```

Una vez llevado cualquiera de ambos procesos, el laboratorio estará disponible para su uso en la plataforma, ya que pertenecería a la lista de imágenes de Docker.

Se puede comprobar el funcionamiento correcto de ambas formas con el comando:

```
$ sudo -u www-data docker images
```

```
sudo: ejecutar un comando como otro usuario.
docker: comando para gestionar Docker.
    images: subcomando para listar las imagenes Docker.
```

Nota El nombre de la imagen Docker será la que se use en el atributo `docker_image_name` del *Laboratorio* de WordPress.

B.2.4. Modificar o eliminar un laboratorio

Cualquier laboratorio se puede modificar a partir de su carpeta correspondiente, modificando su `Dockerfile`. Una vez se ha modificado, se debe reconstruir la imagen Docker del laboratorio para que los cambios surtan efecto, aplicando cualquier de las opciones descritas en la sección anterior.

Nota Recuerde modificar el atributo `docker_image_name` del *Laboratorio* de WordPress si se modifica el nombre de la imagen Docker o esta se elimina.

APÉNDICE C

Código desarrollado

C.1. Scripts para la plataforma

C.1.1. Gestión del tiempo de vida de contenedores

```
#!/bin/bash


# Variables "globales"
CONTENEDORES=$(docker ps -q)     # Lista de contenedores activos (IDs)
VIDA=3600                        # Tiempo de vida de un contenedor (en segundos)


# Mostrar informacion de la ejecucion
echo -e "\e[1mEjecucion actual: $(date +%H:%M:%S)\e[0m\n"
if [ -z "$CONTENEDORES" ]; then
    echo -e "No hay contenedores activos.\n"
else
    echo -e "Hay $(echo $CONTENEDORES | wc -w) contenedores activos.\n"
fi


# Recorrer los contenedores activos
for CONTENEDOR in $CONTENEDORES; do
    # Tiempo de creacion del contenedor (ISO 8601)
    CREACION=$(docker inspect --format '{{ .State.StartedAt }}' $CONTENEDOR)

    INICIO=$(date --date="$CREACION" +%s)       # Tiempo de ejecucion (en segundos)
    AHORA=$(date +%s)                           # Tiempo actual (en segundos)
    DIFF=$(( $AHORA - $INICIO ))                # Tiempo transcurrido (en segundos)

    # Crear informacion del estado de un contenedor
    MSG="$CONTENEDOR:\t$DIFF / $VIDA segundos."

    # Comprobar que el contenedor supero el tiempo de vida
    if [ $DIFF -gt $VIDA ]; then
        echo -e "$MSG \e[31mTiempo de vida alcanzado.\e[0m"

        docker stop $CONTENEDOR > /dev/null

    else
        if [ $DIFF -gt $(( $VIDA - 60 )) ]; then
            echo -e "$MSG \e[33mDestruccion inminente.\e[0m"
        else
            echo -e "$MSG \e[32mDisponible.\e[0m"
        fi
    fi
done

echo
```

Con el fin de optimizar los recursos, la plataforma no tendrá laboratorios activos cuando esta no se esté usando, por lo que se ha creado una tarea de cron que se ejecuta cada minuto y comprueba si hay contenedores activos, y en caso de haberlos, comprueba si alguno ha superado el tiempo de vida establecido (1 hora por defecto) y lo destruye.

El script anterior se encarga de comprobar el tiempo que el contenedor de un laboratorio ha estado activo, y lo destruye cuando este supera el tiempo establecido.

El script realiza las siguientes acciones:

Obtener los contenedores activos Usando el comando `docker ps -q` se obtiene una lista de los contenedores activos, pero solo sus IDs, lo que permite recorrerlos de forma más sencilla; esta información se almacena en la variable `CONTENEDORES`, que se usará para recorrer los contenedores activos.

Para cada contenedor de la lista anterior, se realizan los siguientes procesos:

Obtener el tiempo de creación Usando el comando `docker inspect --format '{{ .State.StartedAt }}'` se obtiene el tiempo de creación de un contenedor en formato ISO 8601, que es el formato que usa Docker para almacenar la fecha y hora de creación de un contenedor; esta información se almacena en la variable `CREACION`.

Obtener el tiempo de ejecución Usando el comando `date --date="$CREACION" +%s` se obtiene el tiempo de ejecución de un contenedor en segundos, y usando el comando `date +%s` se obtiene el tiempo actual en segundos; con estos dos valores se calcula la diferencia entre ellos, que es el tiempo que ha estado activo el contenedor; esta información se almacena en la variable `DIFF` y corresponde a los segundos que han pasado desde que se creó el contenedor hasta el momento actual.

Comprobar si el contenedor debe detenerse Finalmente, se comprueba si el contenedor ha superado el tiempo de vida establecido, y en caso de haberlo superado, se detiene el contenedor usando el comando `docker stop $CONTENEDOR`.

Cabe destacar que el script muestra la información correspondiente por la pantalla, pero la tarea cron redirige su salida a `/dev/null`, por lo que no se mostrará nada por pantalla cuando se ejecute la tarea de cron; sin embargo, el script puede ejecutarse manualmente y observar el estado de los contenedores en cualquier momento.

C.1.2. Actualización de los laboratorios

```
#!/bin/sh


# Variables
carpetas=$(ls -d */ | grep -v '^logs/$' | sed 's#/##')  # Carpetas del repositorio
log=logs/$(date +"%Y-%m-%d %H-%M").log                  # Archivos de registro
duplicada=false                                         # Imagenes duplicadas


# Crear la carpeta de registros
mkdir -p logs                       # Si ya existe 'logs' no hace nada


# Construccion de las imagenes
for carpeta in $carpetas; do

    echo "$carpeta\n-----------------------------------------------" >> $log

    echo "Imagen: $carpeta"


    # Comprobar que la carpeta tiene un Dockerfile
    if [ ! -f $carpeta/Dockerfile ]; then
        echo "    No tiene Dockerfile.\n"
        continue
    fi


    # Eliminar la version anterior de la imagen
    if [ ! -z "$(docker images -q $carpeta)" ]; then
        duplicada=true
        docker rmi $carpeta >> $log 2>&1
    fi


    # Construir la imagen actual
    if [ ! $duplicada = true ]; then
        echo "    Construyendo..."
    else
        echo "    Actualizando..."
    fi

    docker build -t $carpeta $carpeta/ >> $log 2>&1


    # Almacenar la imagen en una lista si se creo correctamente
    if [ ! -z "$(docker images -q $carpeta)" ]; then
        if [ ! $duplicada = true ]; then
            echo "    Creada.\n"
        else
            echo "    Actualizada.\n"
        fi

        images="$images $carpeta"
    fi

    echo "\n" >> $log

    duplicada=false
done


# Mostrar todas las nuevas imagenes generadas
echo "Resumen:"

for image in $images; do
    echo "* $image"
done
```

Los entornos virtualizados de la plataforma están compuestos por imágenes Docker, que son creadas a partir de un fichero `Dockerfile` que contiene las instrucciones necesarias para crear la imagen. Estas imágenes son los laboratorios, y siguiendo la misma estructura planteada en el repositorio de laboratorios, es posible tanto añadir, modificar o eliminar más contenedores a la plataforma.

Técnicamente, no es necesario que los laboratorios pertenezcan al repositorios, lo esencial es que su imagen Docker esté construida en el sistema y disponible para el usuario `www-data`. Sin embargo, el repositorio permite gestionar los laboratorios de forma más sencilla, ya que solo es necesario añadir una carpeta con el nombre del laboratorio y el fichero `Dockerfile` dentro.

El script anterior contenido en la carpeta `labs/` del repositorio, y permite hacer un barrido del mismo y crear aquellos laboratorios que no estén presentes en el sistema, así como actualizar aquellos que hayan sido modificados.

El script realiza las siguientes acciones:

Obtener las carpetas del repositorio Usando el comando `ls -d */` se obtienen todas las carpetas del repositorio, y usando el comando `grep -v '^logs/$'` se filtran todas las carpetas excepto la carpeta `logs/`, que es donde se almacenan los registros de las ejecuciones del script; finalmente, usando el comando `sed 's#/##'` se elimina el carácter `/` de cada carpeta, ya que no es necesario para el script. Esto resulta en una lista de nombres que se usarán para crear las imágenes Docker.

Crear la carpeta de registros Usando el comando `mkdir -p logs` se crea la carpeta `logs/` en caso de que no exista, y en caso de existir, no hace nada. Esto hace que se puedan comprobar los registros de ejecuciones anteriores y los fallos en la construcción de las imágenes Docker de los laboratorios.

Acto seguido, para cada carpeta de la lista obtenida al inicio, se realizan los siguientes procesos:

Eliminar la imagen anterior Usando el comando `docker images -q $carpeta` se comprueba si existe una imagen con el nombre de la carpeta actual, y en caso de existir, se elimina usando el comando `docker rmi $carpeta`; esto produce que se actualice una imagen existente. La imagen se elimina previamente debido a que si se crea una imagen con el mismo nombre que otra, Docker no elimina la anterior, sino que la renombra a `<none>`.

Construir una imagen Usando el comando `docker build -t $carpeta $carpeta/` se construye la nueva imagen Docker usando el fichero `Dockerfile` de la carpeta actual, y se le asigna el nombre de dicha carpeta; esto permite que la imagen tenga el mismo nombre que la carpeta, y por tanto, que sea más fácil de identificar.

Almacenar la imagen en una lista Usando el comando `docker images -q $carpeta` se comprueba si existe una imagen con el nombre de la carpeta actual, lo que indicaría que

la imagen se creó correctamente, y en caso de existir, se añade el nombre de la carpeta a la variable `images`, que se usará para mostrar las imágenes creadas al finalizar la ejecución del script.

Finalmente, se muestran las imágenes creadas en la ejecución del script.

C.1.3. Instalación de Kali Linux usando Vagrant

```
#!/bin/bash


VF_DIR="$HOME/Vagrant/Kali"     # Ubicacion por defecto


# Actualiza la ubicacion por defecto para el Vagrantfile.
# Argumentos:
#   $1: Nueva ruta del Vagrantfile
function define_directory {
    # Se recibe un argumento y es valido
    if [[ $# -eq 1 && -n "$1" ]]; then
        VF_DIR="$1"
    fi

    echo "Ubicacion del Vagrantfile: $VF_DIR"
}

# Instala las dependencias necesarias para la maquina
# virtual, que en este caso son: Vagrant y VirtualBox.
function install_dependencies {
    echo "Instalando dependencias..."

    sudo apt-get update -qq
    sudo apt-get install -y vagrant virtualbox > /dev/null 2>&1

    # Comprobar si se han instalado correctamente
    if [[ $? -eq 0 ]]; then
        echo "Paquetes instalados correctamente."
    else
        echo "Error al instalar los paquetes."
        exit 1
    fi
}

# Crea el directorio para el Vagrantfile.
function build_directory {
    # Directorio para los archivos de Vagrant
    if [[ -d $VF_DIR && -f $VF_DIR/Vagrantfile ]]; then
        echo "El directorio seleccionado no esta vacio."
        read -p "Sobreescribirlo? [s/n]: " answer

        # Comprobar si debe reemplazarse
        if [[ "$answer" != "${answer#[Ss]}" ]]; then
            echo "Reiniciciando el directorio '$VF_DIR'..."
            rm -rf $VF_DIR
        else
            echo "No se puede continuar."
            exit 1
        fi
    else
        echo "Creando el directorio '$VF_DIR'..."
    fi

    mkdir -p $VF_DIR

    # Comprobar si se han creado correctamente
    if [[ $? -eq 0 ]]; then
        echo "Directorio creado correctamente."
    else
        echo "Error al crear el directorio."
        exit 1
```

```
        fi
    }

    # Crea la maquina virtual de Kali Linux usando Vagrant.
    function build_kali {
        cd $VF_DIR

        echo "Creando la maquina virtual de Kali Linux..."
        vagrant init elrey741/kali-linux_amd64  > /dev/null     # Inicializar Vagrant
        vagrant up                                              # Inicializar la MV
    }


    # Inicio del script
    install_dependencies    # Instalar las dependencias
    define_directory $1     # Actualizar la ubicacion de la MV
    build_directory         # Crear el directorio
    build_kali              # Crear la MV de Kali Linux

    echo "Fin de la instalacion."
```

La plataforma también ofrece a los usuarios una forma de instalar una máquina virtual de Kali Linux usando una herramienta llamada Vagrant, que permite crear y gestionar máquinas virtuales de forma sencilla.

El script anterior permite realizar la instalación desde cero, instalando Vagrant y VirtualBox, creando un directorio para la máquina virtual y creando la máquina virtual de Kali Linux usando Vagrant.

Este script se trata de una herramienta de desarrollo para agilizar la ejecución de pruebas; la plataforma ya cuenta con un apartado donde se describen estos mismos pasos, con comandos, para que el usuario pueda replicarlo.

El script realiza las siguientes acciones:

Instalar las dependencias Usando la función `install_dependencies` se instalan las dependencias necesarias para la máquina virtual, que en este caso son Vagrant y VirtualBox.

Usando el comando `apt-get update -qq` se actualiza la lista de paquetes disponibles en el sistema sin mostrar información por pantalla, y usando el comando `apt-get install -y vagrant virtualbox` se instalan los paquetes `vagrant` y `virtualbox`, enviando la salida a `/dev/null` para que no se muestre información por pantalla.

Además, se comprueba si la instalación se ha realizado correctamente usando el código de salida del comando anterior, que es 0 si se ha realizado correctamente, y 1 si ha habido algún error; en caso de error, el script se detiene.

Definir la ubicación del Vagrantfile Usando la función `define_directory` actualiza la ubicación del Vagrantfile por la ruta recibida como argumento `$1` del script; si no se recibe ninguna ruta, se usa la ruta por defecto `$HOME/Vagrant/Kali`.

Crear el directorio Usando la función `build_directory` se crea el directorio donde se almacenará el Vagrantfile, y se comprueba si ya existe un directorio en la ruta seleccionada;

en caso de existir, se pregunta al usuario si desea sobreescribirlo, y en caso de no existir, se crea el directorio.

Además, se comprueba si el directorio se ha creado correctamente usando el código de salida del comando anterior, que es 0 si se ha realizado correctamente, y 1 si ha habido algún error; en caso de error, el script se detiene.

Crear la máquina virtual Por último, usando la función `build_kali` se crea la máquina virtual de Kali Linux usando Vagrant.

Usando el comando `cd $VF_DIR` se cambia el directorio actual al directorio donde se almacenará el Vagrantfile.

Usando el comando `vagrant init elrey741/kali-linux_amd64` se inicializa Vagrant en el directorio anterior, y usando el comando `vagrant up` se levanta la máquina virtual de Kali Linux.

Cabe destacar que el comando `vagrant init` crea un fichero llamado `Vagrantfile` en el directorio actual, que contiene la configuración de la máquina virtual; este fichero se usa para crear la máquina virtual usando el comando `vagrant up`, y este último es necesario para que la máquina virtual se cree y se inicie. No es posible construir la máquina virtual sin iniciarla al menos una vez.

Además, se comprueba si la máquina virtual se ha creado correctamente usando el código de salida del comando anterior, que es 0 si se ha realizado correctamente, y 1 si ha habido algún error; en caso de error, el script se detiene.

C.2. Contenido de los laboratorios

Además de los ficheros *Dockerfiles* y los recursos empleados para cada laboratorio, se han desarrollado 2 proyectos en Python para complementar los laboratorios de OSINT 4.5 y de ransomware 4.12 de la plataforma.

Estos pueden ejecutarse como `./{fichero}.py` añadiendo permisos de ejecución al fichero indicado (`chmod +x {fichero}.py`), o bien de la forma convencional `python3 {fichero}.py`, usando el comando `python3`.

C.2.1. `ip-osint`

Proyecto implementado en el laboratorio de introducción al OSINT 4.5 con el que se puede obtener información de una dirección IP.

```
$ ./main.py -h

usage: main.py [-h] [-i IP] [-l file] [-k file]

IP Information Lookup

optional arguments:
-h, --help            show this help message and exit
-i IP, --ip IP        IP address to check
-l file, --list file  text file with IP addresses to check
-k file, --keys file  JSON file with API keys
```

Listing C.1: Menú de ayuda del proyecto

El proyecto está dividido en 2 secciones principales: los módulos y el script principal.

Módulos Contenidos en la carpeta `utils/`, existe un módulo por cada fuente de datos que se usa para obtener información de una IP, actualmente compuestas de:

- `tor.py`: comprueba si una IP es un nodo de salida de la red TOR.
- `who.py`: obtiene información de la base de datos WHOIS.
- `sho.py`: obtiene información usando la API de Shodan.io.
- `vir.py`: comprueba si una IP está registrada en VirusTotal.
- `geo.py`: obtiene datos de geolocalización y DNS inverso de una IP.

Adicionalmente, el existe un módulo `progressbar.py` que implementa una barra de progreso para la sección de Tor, actualizándose por cada nodo consultado.

Script principal Ubicado en la raíz del proyecto, `main.py`, se encarga de obtener la información de una IP usando los módulos anteriores, y mostrarla por pantalla.

Ejemplo de salida

```
$ ./main.py -i 70.80.0.19

IP: 70.80.0.19

TOR

Analizando 10201 nodos:
[##################################################] 100%

No se encontro informacion sobre la IP.

(...)

whois

Seccion 1/2

NetRange       : 70.80.0.0 - 70.83.255.255
CIDR           : 70.80.0.0/14
NetName        : VL-17BL
NetHandle      : NET-70-80-0-0-1
Parent         : NET70 (NET-70-0-0-0-0)
NetType        : Direct Allocation
OriginAS       : Sin informacion
Organization   : Videotron Ltee (VL-421)
RegDate        : 2004-07-14
Updated        : 2020-10-08
Ref            : https://rdap.arin.net/registry/ip/70.80.0.0

OrgName        : Videotron Ltee
OrgId          : VL-421
Address        : 150 Beaubien West
City           : Montreal
StateProv      : QC
PostalCode     : H2V 1C4
Country        : CA
RegDate        : 2020-05-15
Updated        : 2020-11-02
Comment        : https://www.videotron.com
Ref            : https://rdap.arin.net/registry/entity/VL-421

OrgAbuseHandle : ABUSE263-ARIN
OrgAbuseName   : Network Operations Center
OrgAbusePhone  : +1-514-281-8498
OrgAbuseEmail  : abuse@videotron.ca
OrgAbuseRef    : https://rdap.arin.net/registry/entity/ABUSE263-ARIN

OrgTechHandle  : NOC1226-ARIN
OrgTechName    : Network Operations Center
OrgTechPhone   : +1-514-380-7100
OrgTechEmail   : SRIP@videotron.com
OrgTechRef     : https://rdap.arin.net/registry/entity/NOC1226-ARIN

(...)

Busqueda Inversa de IP

modemcable019.0-80-70.mc.videotron.ca


VirusTotal

La IP no esta registrada en VirusTotal.
```

Listing C.2: Fragmento de la salida del proyecto `ip-osint`.

C.2.2. `stockholm`

Proyecto implementado en el laboratorio de ransomware 4.12 con el que se puede cifrar y descifrar archivos de un directorio concreto de un sistema.

```
$ ./stockholm.py -h

usage: stockholm.py [-h] [-r clave] [-v] [-s] [-p carpeta]

Herramienta casera para 'secuestrar' y 'liberar' ficheros.

optional arguments:
-h, --help  show this help message and exit
-r clave    revierte la infeccion usando la clave de cifrado
-v          version del programa
-s          desactiva la informacion mostrada por pantalla
-p carpeta  descifra todos los ficheros y los almacena en la
   carpeta indicada
```

Listing C.3: Menú de ayuda del proyecto

Se compone de las siguientes funciones:

- `inicializar_analizador()`: Configura y crea el analizador de argumentos de línea de comandos con los que interpretar los valores recibidos.
- `leer_argumentos()`: Lee los argumentos de línea de comandos y devuelve las opciones seleccionadas.
- `validar_fichero(elemento, modo)`: Verifica si un archivo es válido para el cifrado o descifrado.
- `contenido(carpeta)`: Genera una lista con todos los archivos de una carpeta, incluidos los de subcarpetas.
- `secuestrar()`: Cifra los archivos con extensiones específicas en el directorio del script usando el módulo Fernet.
- `liberar(carpeta)`: Descifra los archivos cifrados con extensión `.ft` en el directorio del script y los mueve a la carpeta especificada.

El script funcionará **si y solo si** existe una carpeta llamada `infection/` en la carpeta del usuario del sistema, y los archivos sobre los que se realizará el ataque serán **sola y exclusivamente** aquellos pertenecientes a dicha carpeta.

Si la carpeta no existe, mostrará un error, y si está vacía, no realizará ninguna acción.

Ejemplos básicos

Cifrar todos los archivos de `/root/infection/`.

```
$ ./stockholm.py

Cifrando archivos:
    /root/infection/alpine-standard.iso
    /root/infection/ft_split.c
    /root/infection/libft.h
    /root/infection/libft.pdf
    /root/infection/backup.sql
    /root/infection/nota.txt

Archivos cifrados:
    /root/infection/alpine-standard.iso
    /root/infection/backup.sql
    /root/infection/ft_split.c
    /root/infection/libft.h
    /root/infection/libft.pdf
    /root/infection/nota.txt

    Resumen: 6/6 ficheros descifrados.
```

Listing C.4: Fragmento de la salida del proyecto `stockholm`.

Una vez se han cifrado los ficheros, se puede observar una clave generada en la carpeta `/root/`.

```
$ ls

clave.key  infection  stockholm.py
```

Esta clave se puede utilizar para descifrar los ficheros de `/root/infection/` en la misma carpeta.

```
$ ./stockholm.py -r clave.key

Archivos cifrados:
    /root/infection/alpine-standard.iso.ft
    /root/infection/backup.sql.ft
    /root/infection/ft_split.c.ft
    /root/infection/libft.h.ft
    /root/infection/libft.pdf.ft
    /root/infection/nota.txt.ft

    Total: 6.

Archivos descifrados:
    /root/infection/alpine-standard.iso.ft
    /root/infection/backup.sql.ft
    /root/infection/ft_split.c.ft
    /root/infection/libft.h.ft
    /root/infection/libft.pdf.ft
    /root/infection/nota.txt.ft

Resumen: 6/6 ficheros descifrados.
```

Listing C.5: Fragmento de la salida del proyecto `stockholm`.

APÉNDICE D

Historial de reuniones

A continuación se muestra una lista de las reuniones que tuvieron lugar durante el desarrollo de este trabajo de fin de grado: cada sección corresponde a una reunión y su contenido es un resumen de los cambios realizados antes de dicha reunión.

D.1. 1ª reunión: 12/04/2023

Se crearon los capítulos: Diario D y Borrador 2
El primero, para incluir un historial de mi progreso; el segundo, para subir contenido WIP (*work in progress*) al documento.

Se modificó el diagrama: casos de uso 3.1
Se aplicó un estilo estándar y se mejoraron las dependencias.

Se creó la sección: Arquitectura 3.2
Se incluyeron nuevos diagramas y contenido descriptivo sobre la arquitectura del proyecto.

Se creó la sección: Modelado de Actividades y Transacciones 3.3
Se incluyeron nuevos diagramas y contenido descriptivo de los procesos del proyecto.

Se modificó la sección: Ingeniería de Requisitos 3.1
Se descompusieron los requisitos anteriores en requisitos más atómicos y se han añadido nuevos requisitos no funcionales.

Se creó la sección: Estructura global del proyecto 1.4
Se incluyó una descripción de lo que se plantea realizar con este TFG separándolo en 3 pilares fundamentales.

Se creó la sección: Proceso Creativo 2.2
Se incluyó una descripción de distintos planteamientos sobre la plataforma web que tuvieron lugar durante la investigación previa del TFG.
El nombre de esta sección puede variar, pero al ser un borrador, se elegió algo *llamativo*.

Se inició la creación de un prototipo de la plataforma
Se está configurando una página WordPress (WIP).
Se está practicando con contenedores de Docker (WIP)
Se está tratando de levantar un contenedor a través de la página (WIP).

D.2. 2ª reunión: 03/05/2023

Se añadió una página: contraportada
Se unió el documento PDF de la contraportada del campus virtual al final del documento.

Se modificó la bibliografía: enlace de la ADA [32]
Se sustituyó por una versión corta y equivalente. Ahora todos los enlaces caben en la página.

Se modificó un diagrama: destrucción de una *sandbox* 3.8b
Se añadió un caso de control de errores y el tratamiento del tiempo de vida.

Se modificó una tabla: Wordpress vs Astro vs Drupal 2.1
Se combinaron las tablas Wordpress vs Astro y Wordpress vs Drupal, de forma que aunque se mantienen sus secciones separadas (porque las comparaciones fueron distintas), solo hay una tabla como referencia.

Se modificó la sección: Futuras líneas de trabajo 5.3
Se reformuló el último párrafo, haciendo más hincapié en la posible expansión de la plataforma con más recursos y conceptos fuera del ámbito del pentesting, convirtiéndola no en la temática principal, sino en una de múltiples temáticas.

Se modificó la sección: Jupyter 5.2.2
Se estableció una conclusión para el descarte de esta tecnología en el proyecto.

Se modificó la sección: WordPress 1.5.2
Se añadieron nuevas referencias a la bibliografía.

Se creó un laboratorio: Bypass
Define el concepto de Bypass y la vulnerabilidad CVE-2017-8386 como ejemplo.
El laboratorio contiene dicha vulnerabilidad para explotarla.

Se creó un laboratorio: Fuerza Bruta
Define el concepto de Fuerza Bruta, tipos de ataques más comunes y herramientas.
El laboratorio contiene las herramientas `nmap` e `hydra` para que el usuario pueda obtener las credenciales del usuario administrador del entorno.

Se arregló la sección: Dependencias de contenedores Docker
La sección hacía referencia a *contenedores* de Docker cuando en realidad se trataban de *imágenes*. Ahora la sección cuenta con rigor y se ha optimizado el texto.

D.3. 3ª reunión: 18/05/2023

Se movió el capítulo: Borrador
El capítulo ahora es *Investigación previa* (capítulo 2).

Se investigó la herramienta DVWA
Se plantea la posibilidad de incluir un laboratorio de DVWA en la plataforma.

Se implementa un estado temprano de la plataforma
La plataforma contiene los laboratorios anteriores y se presenta como prototipo.

D.4. 4ª reunión: 15/06/2023

Sección de Alojamiento descrita
Se añadieron imágenes.

Logos de la UMA renombrados y movidos a la carpeta Logos
Ahora están todos los logos juntos.

Secciones Proceso Creativo y Análisis de Tecnologías intercambiadas
Tiene más sentido analizar las tecnologías una vez se ha pasado por un proceso creativo para decidir qué hacer.

Sección de Proceso Creativo mejorada
Más contenido y mejor redactado.

Sección de Metodología mejorada
Más contenido junto a imágenes del proceso.

Contenido de la sección Catálogo mejorado
Se describieron brevemente los conceptos y sus posibles laboratorios.

Contenido de la sección Alojamiento mejorado
Se incluyeron descripciones e imágenes.

Diagramas de Modelado de Actividades y Transiciones mejorados
Se incluyeron nuevos diagramas, más específicos.

Sección de Problemas Encontrados añadida
Describe, guía y documenta los problemas encontrados durante el desarrollo del proyecto así como la toma de decisiones frente a ellos.

D.5. 5ª reunión: 13/07/2023

Se creó un laboratorio: Introducción a Linux
Describe la jerarquía de ficheros y directorios de Linux, el uso de la terminal y comandos divididos en categorías, y la gestión de permisos, grupos y usuarios.
El laboratorio contiene un entorno Linux con grupos y usuarios donde poder practicar.

Se creó un laboratorio: Introducción a Bash scripting
Describe el uso de scripts de Bash, variables, bucles, condicionales, funciones y argumentos.
El laboratorio contiene un entorno Linux con grupos y usuarios donde poder practicar desarrollando scripts con distintos permisos.

Se creó un laboratorio: Introducción a las redes
Describe el uso de redes, protocolos, tipos y tipologías.

Se creó un laboratorio: Análisis de tráfico
Define el concepto de análisis de tráfico y algunas herramientas.
El laboratorio presenta capturas de tráfico para que el usuario pueda analizarlas usando Wireshark.

Se creó un laboratorio: Introducción al OSINT
Describe el concepto de OSINT y cómo buscar información usando distintas herramientas.
El laboratorio presenta una herramienta personalizada, desarrollada por mí, con la que extraer información de una o varias IPs.

Se creó un laboratorio: Ransomware
Define el concepto de ransomware y algunos tipos.
El laboratorio presenta un ransomware personalizado, desarrollado por mí, para que el usuario pueda infectar un entorno seguro -de forma segura- y observar su comportamiento; también permite desinfectar el sistema.

Se aplicaron mejoras a la plataforma
Interfaz de usuario, menú de navegación, registro e inicio de sesión.

D.6. 6ª reunión: 05/09/2023

Se creó un laboratorio: Esteganografía
Define el concepto de esteganografía y muestra algunas herramientas comunes.
El laboratorio presenta una serie de imágenes con infromación oculta con la que el usuario puede llegar a obtener la contraseña de `root`.

Se creó un laboratorio: Hash-cracking
Define el concepto de hash-cracking y las herramientas más comunes.
El laboratorio presenta 3 bases de datos de las que es posible extraer las contraseñas hasheadas de los usuarios.

Se creó un laboratorio: Criptografía
Describe el concepto de la criptografía, sus tipos y ejemplos de algoritmos criptográficos.

Se creó un laboratorio: Escalada de privilegios
Define el concepto de escalada de privilegios y muestra algunos casos de uso.
El laboratorio presenta un entorno Linux con un fichero con permisos débiles que el usuario puede explotar para obtener las credenciales de `root`.

Se aplicaron mejoras a la memoria
Se añadieron nuevas secciones y se mejoró el contenido de las ya existentes.
Se añadieron nuevas referencias a la bibliografía.

Se añadieron los Apéndices
Toma de decisiones, un manual de usuario, código auxiliar y este historial de reuniones.

Se revisó el contenido de los laboratorios
Se aplicaron cambios gramaticales y se corrigieron erratas.

Se revisó el contenido de la memoria
Se aplicaron cambios gramaticales y se corrigieron erratas.

Se añadieron más palabras claves al Resumen
Una palabra clave por cada laboratorio del catálogo (capítulo 4).

Se añadió la página de Agradecimientos

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga