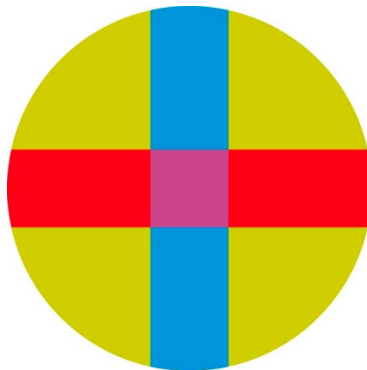


UNIVERSIDAD SAN PABLO - CEU

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA DE SISTEMAS DE INFORMACIÓN



TRABAJO FIN DE GRADO

Desarrollo de una servicio Linux para actualización automática de servicios

Autor: José Manuel Martínez Sánchez

Tutor: Álvaro Sánchez Picot

Junio 2023



UNIVERSIDAD SAN PABLO-CEU

ESCUELA POLITÉCNICA SUPERIOR

División de Ingeniería

Calificación del Trabajo Fin de Grado

Datos del alumno

NOMBRE:

Datos del Trabajo

TÍTULO DEL PROYECTO:

Tribunal calificador

PRESIDENTE:

FDO.:

SECRETARIO:

FDO.:

VOCAL:

FDO.:

Reunido este tribunal el ____/____/_____, acuerda otorgar al Trabajo Fin de Grado presentado por D./Dña. _____ la calificación de _____

Resumen

La creación de este servicio, y por consiguiente esta memoria, ha sido inspirado en los contenidos impartidos en la asignatura *Seguridad Informática y Protección de Datos*, durante el 4º año de carrera. En concreto, se ha inspirado en la vulnerabilidad presentada en el informe de vulnerabilidades creado por la OWASP, Vulnerable and Outdated Components (OWASP Foundation, 2021). Esta vulnerabilidad resalta la criticidad de usar componentes hardware y software actualizados para reducir la posibilidad de sufrir un ciber-incidente.

El objetivo de este proyecto es generar una solución, para usuarios con conocimientos limitados en administración Linux, que permita mantener actualizados los servicios instalados de diversas máquinas de forma automática y sin perjudicar su operatividad. El fin de este servicio es reducir las vulnerabilidades que puedan ser ocasionadas por servicios no actualizados.

En este documento se detallan los procesos efectuados para llevar a cabo el diseño y la implementación de este servicio UNIX para máquinas con sistemas operativos derivados de Debian, SUSE o Fedora.

Palabras Clave

Servicio UNIX, Systemd, Debian, SUSE, Fedora, Python 3, TDD, OWASP

Abstract

The creation of this service, and consequently this memoir, has been inspired by the contents taught in the subject Computer Security and Data Protection, during the 4th year of the course. Specifically, it has been inspired by the vulnerability presented in the vulnerability report created by OWASP, Vulnerable and Outdated Components (OWASP Foundation, 2021). This vulnerability highlights the criticality of using up-to-date hardware and software components to reduce the possibility of cyber-attacks.

The objective of this project is to generate a solution, for users with minimal knowledge in Linux administration, that allows keeping the installed services of different machines up to date automatically and without damaging their operability. The purpose of this service is to reduce vulnerabilities that may be caused by unpatched services.

This document details the processes carried out to design and implement this UNIX service for machines with operating systems derived from Debian, SUSE or Fedora.

Keywords

UNIX Service, Systemd, Debian, SUSE, Fedora, Python 3, TDD, OWASP

Índice de contenidos

Capítulo 1 Introducción	1
1.1 Contexto.....	1
1.2 Objetivos del proyecto.....	2
1.3 Contenidos de la memoria	3
1.4 Terminología básica del proyecto	4
Capítulo 2 Gestión del proyecto	7
2.1 Modelo de ciclo de vida	7
2.2 Papeles desempeñados en el proyecto	8
2.3 Planificación	9
2.4 Presupuesto	10
2.5 Ejecución	11
Capítulo 3 Análisis	13
3.1 Introducción del análisis	13
3.2 Análisis de herramientas similares.....	13
3.3 Justificación de las tecnologías de uso.....	14
3.4 Especificación de requisitos	16
3.5 Análisis de los casos de uso y de las clases de análisis.....	19
3.6 Análisis de seguridad	22
Capítulo 4 Diseño e implementación	24
4.1 Arquitectura del sistema.....	24
4.2 Modelo de clases de diseño	29
4.3 Diseño físico de datos	31
4.4 Diseño de la interfaz de usuario	34
4.5 Entorno de construcción.....	36
4.6 Plan de pruebas	37
4.7 Diagrama de infraestructuras de nivel 3	44
Capítulo 5 Construcción	45
5.1 Referencia al repositorio de software.....	45
5.2 Manuales	45

5.2.1	Módulos necesarios	45
5.2.2	Instalación del servicio	46
5.2.3	Configuración del servicio y ejecución	46
5.2.4	Desinstalación del servicio	47
5.2.5	Acceso al dashboard	47
5.2.6	Ejecución de test unitarios	48
Capítulo 6 Conclusiones y líneas futuras		49
Capítulo 7 Bibliografía		53

Índice de ilustraciones

Ilustración 2.1.1 – Estructura Modelo de Ciclo de Vida del Proyecto	8
Ilustración 2.3.1 - Diagrama Gantt con la planificación Inicial del proyecto	10
Ilustración 3.5.1 - Análisis Casos de Uso.....	19
Ilustración 3.5.2 - Diagrama de ficheros.....	21
Ilustración 4.1.1 - Arquitectura del sistema	24
Ilustración 4.2.1 - Modelo de clases de diseño	30
Ilustración 4.4.1 - Página principal dashboard	34
Ilustración 4.4.2 - Datos de las máquinas con S.O. derivado de Debian	35
Ilustración 4.4.3 - Datos de máquinas Debian y Fedora en periodo específico....	35
Ilustración 4.6.1 - Escenario para prueba de servicio.....	42
Ilustración 4.6.2 - Dashboard ejecución prueba escenario	43
Ilustración 4.6.3 - Datos brutos ejecución prueba modelo	43
Ilustración 4.7.1 - Diagrama infraestructura nivel 3	44
Ilustración 5.2.1 - Resultado instalación servicio	46
Ilustración 5.2.2 - Levantamiento del servicio.....	47

Índice de tablas

Tabla 1 - Comandos necesarios para el servicio y su función.....	28
Tabla 2 - Descripción de los datos necesarios para la ejecución	32
Tabla 3 - Descripción de los datos resultantes de la ejecución	33
Tabla 4 - Listado de test unitarios implementados	41

Capítulo 1

Introducción

1.1 Contexto

En la actualidad, el concepto de ciberseguridad se encuentra soportado por lo que se conocen como los cinco pilares de la ciberseguridad. Estos pilares, promovidos por el Departamento de Defensa de los Estados Unidos (Departamento de Defensa de los E.E.U.U, 2023), resaltan los métodos conocidos que los ciberdelincuentes disponen para afectar a la seguridad e integridad de nuestras máquinas y de nuestros datos. Además, en estos se detallan sus causas y se definen los métodos que existen actualmente para limitar su impacto. Estos 5 pilares son conocidos como: Confidencialidad, Integridad, Disponibilidad, Autenticidad y No Repudio.

Uno de estos pilares, el pilar de la **Integridad**, se enfoca en el efecto perjudicial que la manipulación y/o modificación de cualquier tipo de software o hardware, por parte de un ciberdelincuente puede generar (KeepCoding, 2022). Dentro de este pilar es donde se desarrolla la causa cuyo objetivo, para este proyecto, es limitar. Esta causa se conoce como **vulnerabilidades en los dispositivos debido a la falta de mantenimiento y actualización de sus servicios**, la cual se quiere limitar en máquinas UNIX.

Como se puede observar en los datos presentados por el Instituto Nacional de Ciberseguridad o INCIBE, durante el año 2022, en España se detectaron un total de **3.309.302 de dispositivos vulnerables** y 26.431 nuevas vulnerabilidades documentadas. Estas vulnerabilidades, junto con otras formas de ciberintrusión, produjeron un total de 118.820 incidencias entre empresas y ciudadanos (INCIBE, 2022). Solo en el sector empresarial, donde se dan 9 de cada 10 incidentes

relacionados con sistemas vulnerables, detectar o no este tipo de vulnerabilidades puede suponer un factor final en la continuidad de cada empresa.

Para las grandes empresas, el análisis y la remediación de estas vulnerabilidades no supone un desafío significativo en el ejercicio de sus prácticas. Esto se debe a que estas empresas, gracias a su alto presupuesto, pueden contar con equipos de seguridad altamente cualificados y herramientas de seguridad especializadas en el seguimiento de vulnerabilidades, que facilitan su labor.

Sin embargo, para el resto de los usuarios o empresas que requieren el uso de las tecnologías, pero no cuentan con capital humano cualificado ni con las medidas de seguridad apropiadas debido a su presupuesto, realizar el seguimiento de estas vulnerabilidades puede llegar a ser una tarea imposible.

En España, este grupo de empresas se encuentran dentro de lo que se conoce como las Pymes (Pequeñas y Medianas Empresas), siendo estas el 99,8% del total de las empresas en España (Ministerio de Industria, Comercio y Turismo, 2022).

1.2 Objetivos del proyecto

Es por ello que, para este proyecto se ha planteado el desarrollo un servicio UNIX cuya función es la de analizar y actualizar el versionado de los servicios instalados en máquinas Linux, con el objetivo de limitar el riesgo producido por este tipo de vulnerabilidades.

Gracias a este proyecto, se pretende que los servicios de cualquier sistema Linux, derivados de SUSE, Debian o Fedora, se mantengan actualizados a la última versión retrocompatible, sin la intervención del usuario en la máquina. Aparte, se desea proveer al servicio de herramientas visuales que permitan, a los usuarios no especializados, obtener una imagen real del riesgo de sus activos.



Además, existen otros objetivos secundarios de igual relevancia que son:

1. Diseñar un servicio sencillo e intuitivo, que pueda ser instalado y ejecutado por cualquier usuario con conocimientos medios de Linux.
2. Desarrollar un servicio seguro el cual no comprometa la integridad, la disponibilidad y la accesibilidad de las máquinas involucradas en el análisis del servicio.
3. Generar unas métricas, en base a los resultados del análisis, e implementar una forma de representación visual con el cual el usuario pueda conocer la situación actual de los servicios instalados en las máquinas.
4. Implementar una serie de pruebas funcionales con las cuales garantizar el cumplimiento de los requisitos y el funcionamiento del servicio.

1.3 Contenidos de la memoria

A lo largo de este documento, se exponen detalladamente los procesos llevados a cabo para la realización del proyecto. Estos procesos se encuentran divididos en varios capítulos que son:

2. Gestión del proyecto, especificando el modelo de ciclo de vida en el que se basa la gestión del proyecto, la planificación desarrollada en base al modelo y el presupuesto estimado para la ejecución del proyecto. Además, se reflexiona sobre el resultado de la planificación y los gastos cometidos, tras la finalización del proyecto.

3. Planificación del servicio, definiendo los requisitos funcionales y no funcionales del servicio, detallando cada caso de uso y analizando su impacto desde el punto de vista de la seguridad.

4. Diseño del servicio, detallando la arquitectura, especificando el diseño de clases, el formato de los datos y la interfaz de usuario. Aparte, se

enumeran los entornos usados para su construcción y se detalla el plan de pruebas implementado para la validación de los requisitos.

5. Servicio resultante, referenciando el código desarrollado y detallando los manuales descritos para su correcta instalación, configuración y uso.

6. Conclusiones, reflexionando sobre la relevancia intrínseca de la materia del proyecto e identificando los posibles errores cometidos durante el desarrollo. Por último, se detallan las posibles líneas futuras de desarrollo que podría tener el proyecto.

7. Bibliografía, citando todas las fuentes consultadas para el análisis del proyecto y el desarrollo del servicio.

1.4 Terminología básica del proyecto

Para comprender los contenidos de esta memoria, se ha desarrollado la siguiente lista, en la cual se definen varios términos fundamentales en la materia de este proyecto. Estos son:

- **Ciberataque.** Según IBM, los ciberataques son intentos no deseados de robar, exponer, alterar, inhabilitar o destruir información mediante el acceso no autorizado a los sistemas (IBM, 2023).
- **Ciberdelincuente.** Según el Instituto Nacional de Ciberseguridad, un ciberdelincuente es aquella persona que busca sacar beneficio de los fallos de seguridad, utilizando para ello distintas técnicas como es la ingeniería social o el malware (INCIBE, 2023).
- **Servicio UNIX.** Un servicio es un programa que se ejecuta en segundo plano, fuera del control interactivo de los usuarios del sistema, ya que carecen de una interfaz. Esto con el fin de proporcionar aún más seguridad, pues algunos de estos servicios son cruciales para el funcionamiento del sistema operativo (Deyimar, 2022).



- **Retrocompatibilidad.** La retrocompatibilidad (a veces conocida como compatibilidad hacia atrás) es una propiedad de un sistema, producto o tecnología que permite la integración con un sistema heredado más antiguo o con una entrada diseñada para dicho sistema, especialmente en telecomunicaciones e informática. (Mandge, 2021)
- **Versionamiento semántico.** El versionamiento semántico es un convenio o estándar a la hora de definir la versión del código, dependiendo de la naturaleza del cambio que estás introduciendo (Armesto, 2023). En este versionado se identifican 4 cambios cuya estructura es Major.Minor.Patch-Beta, y se definen como:
 - Major: Cambio drástico en el software que no es compatible con código realizado en versiones anteriores.
 - Minor: Cambio que añade alguna característica nueva al software o modifica alguna ya existente, pero que sigue siendo compatible con código existente.
 - Patch: Cambio destinado a eliminar errores en el código, siendo el cambio retrocompatible.
 - Beta: Igual que el Patch, se trata de un cambio destinado a corregir bugs, siendo el cambio retrocompatible.
- **CVE.** Los puntos vulnerables y las exposiciones comunes (CVE) conforman una lista de las fallas de seguridad informática que está disponible al público (RedHat, 2021).

Capítulo 2

Gestión del proyecto

2.1 Modelo de ciclo de vida

Para el desarrollo del servicio en el que está enfocado el proyecto, se ha decidido como modelo de ciclo de vida el modelo en cascada o clásico (IONOS, 2019). Este modelo se basa en la realización de las etapas del desarrollo del producto de forma lineal, de tal manera que solo se podrá empezar con una fase sí y sólo si la fase anterior ha sido finalizada. La decisión de usar este modelo se basa en varias de las ventajas que ofrece, de las cuales el desarrollo del proyecto y las personas involucradas pueden beneficiarse. Estas ventajas son:

Definición de una estructura clara. El modelo permite establecer una estructura simple, permitiendo crear un único flujo continuo de procesos definidos. Este flujo reduce las probabilidades de que alguna parte del proyecto se quede sin terminar y, en el supuesto de que se produzca un obstáculo, este podrá ser resuelto de inmediato por los participantes del proyecto.

Difusión adecuada de la información. Al establecer un proceso sumamente metódico, el intercambio de información entre ambas partes mejora con respecto al uso de otras metodologías.

En la siguiente ilustración (Ilustración 2.1.1) se puede observar en mayor detalle, el modelo seleccionado para la realización de este proyecto, especificando los distintos procesos que van a ser necesarios para cometer el proyecto.

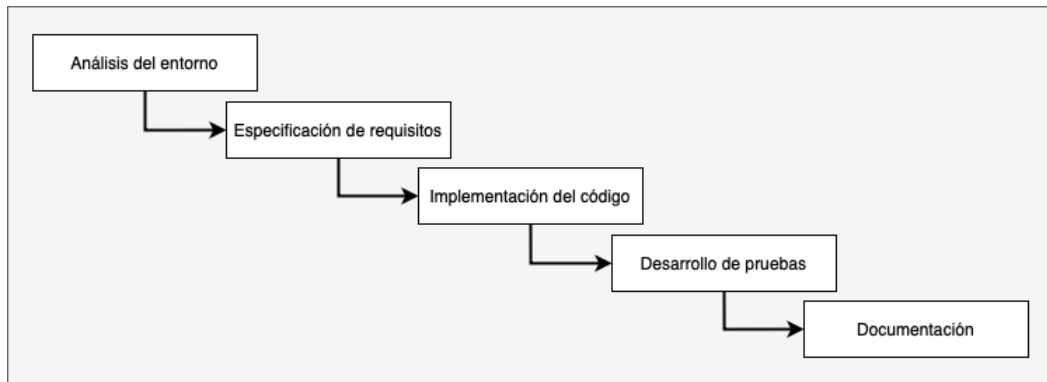


Ilustración 2.1.1 – Estructura Modelo de Ciclo de Vida del Proyecto

2.2 Papeles desempeñados en el proyecto

Los personajes que formarán parte activa en el desarrollo del servicio serán el tutor del TFG (también conocido como el dueño del producto); y el estudiante que, debido al propósito del proyecto, realizará las funciones de diseñador, programador y tester del servicio.

Debido a que los conocimientos del estudiante en la materia, adquiridos en durante la carrera; y a los conocimientos del tutor, no se ha considerado necesario la involucración de algún usuario experto para la resolución del problema. Aunque, en el caso de que sea necesaria la involucración de un usuario experto, debido a algún impedimento que se pueda dar, se tiene conocimiento de varios profesores con elevados conocimientos en la materia que puede ayudar.

Durante la ejecución del proyecto, el dueño del producto tendrá como función validar cada uno de los hitos a realizar. Para superar con éxito cada uno de los hitos, se deberán establecer unos parámetros de validación previos a la ejecución del proyecto entre ambas partes.

En el caso del estudiante, sus funciones serán analizar el entorno para detectar otras posibles soluciones, establecer los requisitos y herramientas a usar, implementar una solución que cumpla con los requisitos. Aparte, el usuario debe generar una serie de pruebas funcionales, que demuestren al dueño del proyecto



el correcto funcionamiento del servicio. Por último, será responsable de actualizar el estado del proyecto al tutor, cada vez que se finalice con una de las tareas impuestas en la planificación, a través de una reunión.

2.3 Planificación

Siguiendo con la metodología definida (2.1 Modelo de ciclo de vida), se ha diseñado una planificación en cascada dividida en 5 pasos o hitos. Estos hitos son:

1. **Análisis previo del entorno.** Antes de establecer los requisitos del servicio, se realiza un estudio de mercado en busca de algún producto o servicio que realice una función igual o similar a la del proyecto. Además, se estudian las diferentes tecnologías que pueden ser usadas para desarrollar el producto.
2. **Especificación de los requisitos.** Tras haber realizado el análisis, se establecen los requisitos funcionales y no funcionales que debe cumplir el servicio. Estos requisitos han de ser validados con el dueño del proyecto.
3. **Implementación de la solución.** Una vez definidos los requisitos, se plantea una arquitectura que cumpla con los requisitos marcados y se desarrolla una solución funcional.
4. **Desarrollo de pruebas.** Cuando se haya finalizado la implementación de la solución, se generará una batería de pruebas y un entorno de simulación, para poder comprobar el funcionamiento del servicio y el cumplimiento de los requisitos.
5. **Documentación y despliegue.** Por último, se despliega el software desarrollado, en conjunto con la documentación necesaria. Esta documentación debe incluir tanto la memoria del proyecto, como un manual de uso de la herramienta.

Además de los hitos previos, se incluye una tarea colchón como margen de maniobra, brindando cierta flexibilidad en el tiempo de ejecución del proyecto en el caso de que se supere el tiempo establecido en otra de las tareas previas. Esta

tarea tendrá un valor, en días, equivalente al 10% del tiempo total esperado para realización de las tareas descritas.

Para poder observar con mayor detalle la planificación se adjunta el siguiente diagrama de Gantt (Ilustración 2.3.1), en el que se detallan los hitos establecidos y el tiempo estimado de realización de cada uno.

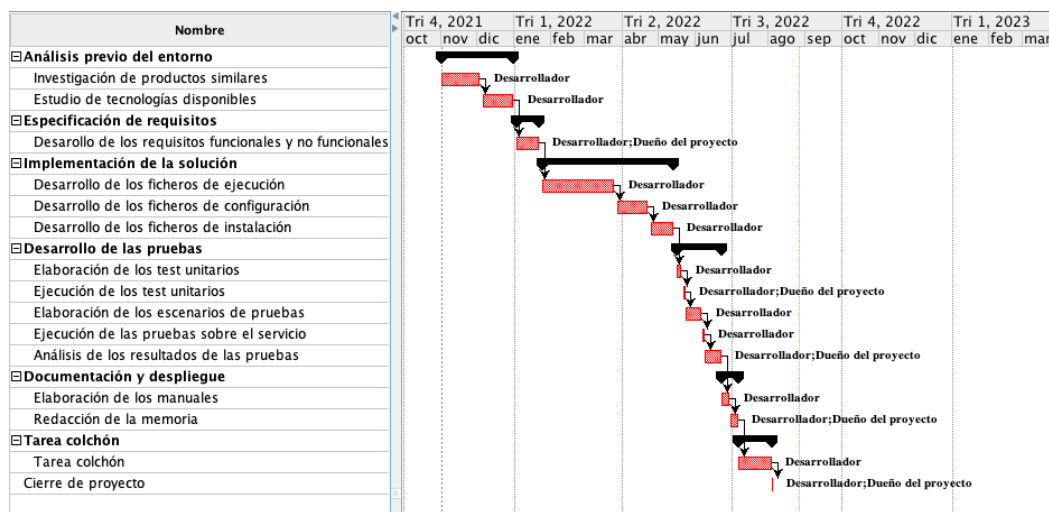


Ilustración 2.3.1 - Diagrama Gantt con la planificación Inicial del proyecto

Tras la finalización del proyecto, se comprueba que la planificación desarrollada al principio no se ha cumplido debido a la falta de entregables presentados por parte del autor del proyecto. La falta de entregables obligó a que la presentación del proyecto se realizará un año después de la fecha estimada. Durante el tiempo extra, se trabajó con el tutor en la mejora del servicio y su memoria.

2.4 Presupuesto

Tras haber realizado evaluación preliminar del entorno y haber considerado las herramientas que van a ser necesarias para ejecutar el proyecto, se estima que el presupuesto necesario para la creación del servicio será de 250 euros.



Se ha considerado un presupuesto tan bajo al no ser necesario realizar un desembolso económico inicial para la adquisición de componentes hardware (ordenador, periféricos, conexión a internet, etc.) y software (IDEs, máquinas virtuales, etc.) necesarios para la implementación del servicio. Esta carencia de desembolso se debe a que el estudiante ya cuenta con estos componentes.

Aún así, el presupuesto valorado de 250 euros se reserva para la realización de las pruebas, en el caso de que la capacidad actual de hardware disponible no permita ejecutarlas. De ser así, se hará uso de la infraestructura tipo AWS (AWS, 2022) para lanzar diversas máquinas virtuales con las que realizar las pruebas sobre los diferentes sistemas operativos que se encuentran dentro del alcance del proyecto.

En el supuesto de que el desarrollador del servicio no contase con ninguno de los componentes necesarios para la ejecución de este proyecto, se estima que el desembolso necesario sería superior a 2.000 euros. Este nuevo presupuesto servirá para adquirir los elementos descritos previamente.

2.5 Ejecución

A lo largo del desarrollo del proyecto, se han dado diferentes contratiempos que han impactado tanto en la planificación (2.3 Planificación) como en el presupuesto original del proyecto (2.4 Presupuesto). Estos inconvenientes han surgido como resultado del modelo de ciclo de vida elegido, ya que no se supieron identificar diversas desventajas, principalmente relacionadas con el flujo de trabajo lineal del modelo.

A raíz de este flujo a este flujo, no se pudieron detectar varios problemas ligados a la obtención de las versiones de los servicios para las diferentes distribuciones durante la **Implementación de la solución**, hasta que se alcanzó el **Desarrollo de las pruebas**. Por este motivo se tuvo que volver al proceso previo, rompiendo su enfoque lineal, y corregir los errores detectados. Además, después de resolver los

problemas, fue necesario refactorizar las pruebas del servicio para satisfacer las nuevas modificaciones, con el fin de asegurar su validez.

Este inconveniente afectó a los tiempos destinados para la realización de ambos hitos, pero no influyó en la fecha final del proyecto. Esto se debió a la creación de la tarea colchón, la cual pudo amortizar el tiempo perdido con el tiempo asignado a esta.

En cuanto al presupuesto, sí que se ha hecho uso del capital establecido para las pruebas del servicio, pero este no ha superado al presupuesto original del proyecto. Como se indicó en el presupuesto, los fondos fueron destinados al despliegue de varias máquinas virtuales en AWS con las distribuciones incluidas en el alcance para ejecutar las pruebas diseñadas.

Capítulo 3

Análisis

3.1 Introducción del análisis

Previamente al planteamiento y diseño de los requisitos del servicio, se ha considerado conveniente realizar un estudio de mercado en busca de herramientas que compartan objetivos similares al planteado. Aparte, realizará un análisis y justificación del uso de las herramientas técnicas que serán fundamentales para la realización del proyecto.

3.2 Análisis de herramientas similares

Actualmente, existen diferentes herramientas o servicios en el mercado, que cuentan con distintas funciones para gestionar el mantenimiento de las versiones de los servicios instalados en distribuciones Linux. Entre estas herramientas y servicios analizados, destacan las siguientes por ser las mas equiparables a los objetivos del proyecto.

En el caso de las herramientas, la que más destaca es **Qualys** y en especial su módulo dedicado para la gestión de vulnerabilidades, denominado Vulnerability Management (Qualys Enterprise, 2021). Esta herramienta dispone de diversas funciones, entre las que se encuentran la realización de escaneos de seguridad basados en la lista CVE (The MITRE Corporation, 2022), o la actualización de servicios con vulnerabilidad conocida. Sin embargo, existen dos hándicaps que dificultan la implementación y uso de esta herramienta, que son:

- El precio de adquisición de la herramienta. La subscripción a este servicio puede suponer al usuario un desembolso de entre \$295 y \$1.995 anuales

por máquina (Cybersecurity Pricing, 2018), dependiendo del número de máquinas que requieran disponer de este módulo.

- La dificultad técnica para su implementación. Además del precio de la herramienta, este no incluye el sueldo de un ingeniero de seguridad, el cual será imprescindible para la implementación y gestión la herramienta. Siendo esto debido a la complejidad técnica de la misma y a la cantidad de recursos que elegir entre los que dispone.

En el caso de los servicios, se encuentra como mejor solución el paquete **unattended-upgrades** (Debian Foundation, 2021), el cual te permite de forma automática actualizar los paquetes de los servicios de una única máquina. Esta una solución es muy similar a la planteada en este proyecto, sin embargo, se observan diversas limitaciones que comprometen su usabilidad, que son:

- Complejidad de configuración. Para aquellos usuarios sin conocimientos en administración Linux, la puesta en marcha del servicio puede ser tediosa al no contar con una interfaz amistosa y al no disponer de mucha documentación del servicio.
- Limitación de análisis. Este servicio, siendo únicamente posible ejecutarlo en local, se encuentra exclusivamente disponible para aquellas distribuciones derivadas de Ubuntu.
- Ineficiencia en la actualización de los servicios. El servicio no toma en cuenta la retrocompatibilidad entre las versiones de los paquetes de los servicios instalados y actualizados. Además, este servicio no dispone de la capacidad de reiniciar los servicios una vez hayan sido actualizados.

3.3 Justificación de las tecnologías de uso

Al ser la finalidad de este proyecto el desarrollo de un servicio para unas distribuciones concretas de Linux, únicamente se considera necesario justificar

dos tecnologías para la implementación del servicio, el lenguaje de programación y el administrador de servicios para la ejecución de este.

Lenguaje de programación. Para este proyecto se ha decidido usar como lenguaje de programación Python 3 (Python Software Foundation, 2022) debido a tres factores fundamentales, que son:

- Existencia de un gran número de módulos ya implementados. Algunos de los requerimientos necesarios para la creación del servicio ya se encuentran implementados por otros desarrolladores como módulos instalables. Por lo tanto, no es necesario crearlos desde cero, reduciendo así el tiempo de desarrollo y limitando posibles errores al haber sido ya testeados.
- Facilidad de desarrollar y testear una aplicación de tamaño reducido. Este lenguaje nos permite desarrollar y ejecutar aplicaciones sin la necesidad de crear una gran estructura de clases o ficheros. Por ese motivo, para la implementación de este servicio que no se espera que sea muy grande, nos permite obtener una mayor flexibilidad en cuanto a la estructura del código.
- Gran documentación disponible para el desarrollo de un servicio. Gracias a la información disponible en libros y documentación web, diseñar un servicio que ejecute por debajo de una aplicación de Python 3 es muy sencillo y rápido. Ya que para realizarlo solo se requieren de algunos documentos necesarios de configuración.

Administrador de servicios Linux. Para este proyecto se ha decidido usar como administrador de servicios de Linux Systemd (Systemd, 2022), ya que este es el administrador preinstalado en todas las distribuciones dentro del alcance del proyecto. Gracias a este administrador, el usuario podrá ejecutar y detener el servicio fácilmente cuando él lo deseé.

3.4 Especificación de requisitos

Los requisitos establecidos por los participantes del proyecto para el desarrollo del servicio son:

Requisitos funcionales

- RF1. El servicio debe poder ser ejecutado desde cualquier máquina con una distribución derivada de UNIX que disponga de Python 3 y de la herramienta de gestión de servicios systemctl.*
- RF2. El servicio debe tener la capacidad de realizar sus funciones de conexión, análisis y actualización de servicios únicamente sobre máquinas Linux cuya distribución sea derivada de Fedora, SUSE y Debian.*
- RF3. El servicio debe realizar las conexiones con las máquinas analizadas a través del servicio SSH.*
- RF4. El servicio deberá extraer los servicios y versiones de los mismos instalados en una máquina cuya distribución se encuentre dentro de la incluidas en el proyecto, y la cual disponga del gestor de paquetes predefinido para esa distribución.*
- RF5. El servicio deberá tener la capacidad de actualizar automáticamente los servicios de una máquina a su última versión disponible, si se comprueba que la versión instalada es retrocompatible con la versión a instalar. Para cerciorarse que dos versiones son retrocompatibles, la versión Mayor en ambas versiones debe de ser igual.*
- RF6. En los casos donde los servicios no sean retrocompatibles con la última versión disponible o no se pueda demostrar con certeza, el servicio no deberá actualizarlos. Esto es independientemente de que existan actualizaciones retrocompatibles con la versión instalada.*



- RF7. *El servicio deberá generar y enviar las instrucciones necesarias para la extracción de información y la actualización de los servicios con independencia del sistema operativo de las máquinas analizadas.*
- RF8. *El servicio podrá ser ejecutado sobre la máquina local, donde el servicio es ejecutado, o sobre una o más máquinas remotas. Si estas máquinas cuentan con los sistemas operativos disponibles en el alcance y disponen de la capacidad de realizar conexiones remotas.*
- RF9. *El servicio realizará sus operativas de forma automática, sin la intervención activa del usuario. El usuario solo debe introducir las máquinas analizadas y las credenciales de acceso a los mismos. Siendo las credenciales iguales para todas las máquinas analizadas.*
- RF10. *El servicio será capaz de repetir el análisis de las máquinas cada cierto tiempo. Este parámetro dispondrá de un valor en tiempo predefinido, pero el usuario deberá tener la capacidad de poder modificarlo.*
- RF11. *El servicio podrá realizar el análisis de las máquinas sobre una o más direcciones IP, rangos de direcciones IP o redes completas.*
- RF12. *En ningún momento el servicio debe ser deshabilitado en caso de error. En el caso de que surja uno, el servicio debe continuar con sus operativas. Solo se deshabilitará el servicio si el usuario lo desea.*
- RF13. *El servicio se deberá instalar en la máquina mediante un fichero de configuración. La ejecución de este será suficiente para que el usuario pueda usarlo.*
- RF14. *El servicio deberá contar un fichero de configuración que permita la desinstalación del servicio. La ejecución de este será suficiente para que se eliminen todos los ficheros del servicio.*
- RF15. *El servicio debe almacenar distintos datos resultantes de la ejecución del servicio. Estos datos deben ser almacenados en ficheros CSV y, en ningún momento, deben guardar información sensible con la cual se pueda identificar el direccionamiento de una máquina.*

- RF16. El servicio dispondrá de un dashboard con el cual el usuario podrá observar datos relevantes sobre el estado del versionado de las máquinas analizadas. Los datos de este servicio provendrán de los datos almacenados de la ejecución del servicio.*
- RF17. El dashboard solo estará disponible cuando el servicio se encuentre habilitado. En caso contrario, este no podrá ser accedido.*
- RF18. El dashboard debe de contar con parámetros de configuración con el fin de que el usuario pueda obtener información más precisa de una métrica que se obtiene mediante la ejecución del servicio.*
- RF19. Cada vez que el servicio sea habilitado la información previa será almacenada en un nuevo fichero cuyo nombre debe de identificar la fecha y fecha de creación del documento.*
- RF20. Cada vez que el servicio sea habilitado la nueva información será almacenada en el fichero destinado al resultado de la ejecución. La información previa que haya en ese documento debe de ser borrada.*

Requisitos no funcionales

- RNF1. El usuario deberá tener la capacidad de introducir el direccionamiento de las máquinas mediante direcciones IP, rangos de direcciones IP o redes completas.*
- RNF2. Los ficheros que no sean parte del funcionamiento del servicio (ficheros de instalación, configuración y resultados), tendrán únicamente permisos de lectura y escritura para el usuario responsable de la ejecución.*
- RNF3. Los ficheros que sean parte del funcionamiento del servicio (scripts), tendrán únicamente permisos de ejecución para el usuario responsable de la ejecución.*
- RNF4. Los usuarios, que no sean responsables de la ejecución del servicio, no contarán con permisos de lectura, escritura y ejecución de ninguno de los ficheros que sean parte del servicio.*



RNF5. El tiempo de análisis de una máquina debe ser inferior a los 5 minutos y debe tener la capacidad de analizar 2 máquinas como mínimo.

3.5 Análisis de los casos de uso y de las clases de análisis

Habiendo definido los requisitos del servicio (3.4 Especificación de requisitos) y tras realizar un estudio sobre las diferentes casuísticas de uso que se darán en el proyecto, se han determinado los casos de uso existentes que requieren de la involucración del usuario (Ilustración 3.5.1).

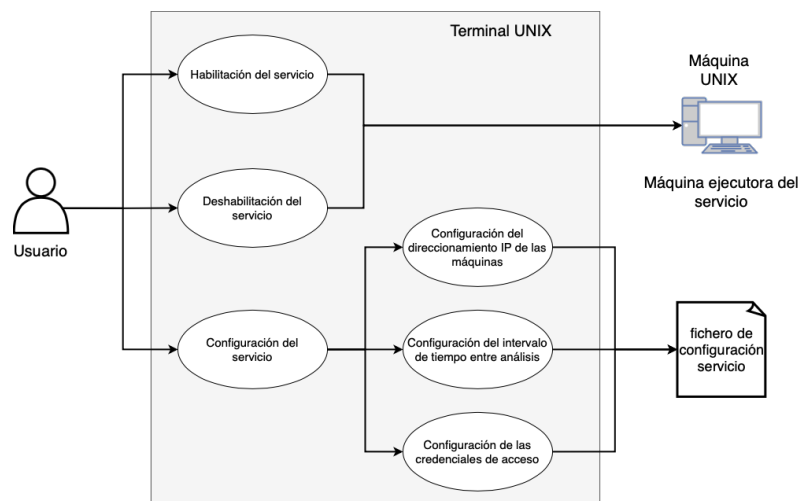


Ilustración 3.5.1 - Análisis Casos de Uso

Caso Nº1 – Configuración de servicio. Para que el servicio pueda ser ejecutado, este debe de ser previamente instalado y configurado. Para ello, el usuario debe ejecutar en la máquina el fichero de instalación. La ejecución de este copiará los ficheros del servicio en su directorio objetivo de la máquina, habilitando así el uso del servicio. Una vez instalado, se deberán establecer los valores de configuración del análisis, situados en un fichero de configuración, para poder establecer el direccionamiento, las credenciales de acceso y el tiempo entre escaneos.

Caso Nº2 – Habilitación o deshabilitación del servicio. Al tratarse de un servicio, el usuario solo deberá tener la capacidad de habilitar o deshabilitar el mismo, mediante el comando *systemctl*. Además, cuando el sistema esté habilitado, el usuario deberá poder acceder a un dashboard con datos generados en la ejecución del servicio.

Caso Nº3 – Desinstalación del servicio. Cuando ya no desee usar la herramienta, el usuario podrá eliminar el servicio del sistema ejecutando un fichero de desinstalación. Este debe de parar el servicio y eliminar todos los ficheros que pertenezcan al mismo.

La ejecución de estos casos será independientemente del sistema operativo de la máquina donde se ejecute el servicio, siempre que esta cumpla con los requisitos definidos.

A partir de los casos de uso extraídos, se ha obtenido como resultado el siguiente modelo de clases con el que se definirá una primera estructura del proyecto. Esta estructura cuenta con **8 tipo de ficheros** en los que se incluyen:

- **Instalador:** fichero cuya función es copiar los ficheros del proyecto a la estructura de directorios del sistema, para poder ejecutar el servicio.
- **Desinstalador:** fichero cuya función es eliminar los ficheros del proyecto, tanto los originales como aquellos derivados de la ejecución del servicio, de la estructura de directorios del sistema.
- **Fichero de configuración:** fichero de configuración de los parámetros del servicio, a incluir por el usuario y necesarios para la ejecución.
- **Fichero de instanciación del servicio:** fichero de configuración para la máquina, necesario para que la aplicación desarrollada sea reconocida como un servicio (Galvez, 2022).
- **Fichero de almacenamiento de los resultados:** fichero en donde se almacenarán las métricas extraídas a raíz de la ejecución del servicio.



- **Script de configuración del servicio:** script donde se localizan los métodos desarrollados.
- **Script aplicación del servicio:** script donde se instancian los métodos desarrollados y se ejecuta el servicio.
- **Script creación del dashboard:** script encargado de instanciar y generar el dashboard interactivo con los datos obtenidos de la ejecución del servicio.

Las conexiones entre estos ficheros se pueden ver representados en el siguiente diagrama de ficheros (Ilustración 3.5.2).

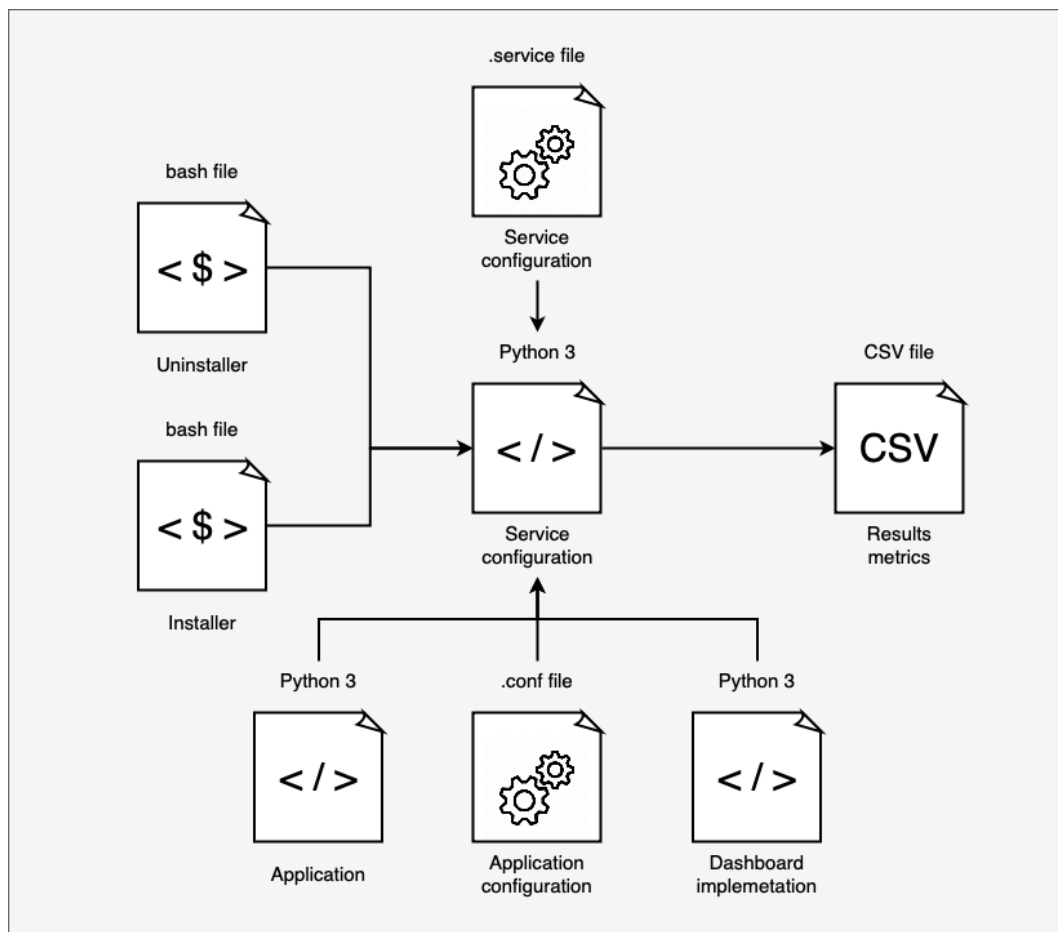


Ilustración 3.5.2 - Diagrama de ficheros

3.6 Análisis de seguridad

Desde el punto de vista de la seguridad existen diferentes riesgos que pueden afectar a la confidencialidad, integridad y/o disponibilidad de las máquinas participantes en el servicio. De los riesgos analizados, se han detectado 3 posibles amenazas que pueden afectar al servicio y a las máquinas analizadas.

La integridad y confidencialidad de la información que se puede obtener de las máquinas a través de la configuración del servicio. Para que se puedan analizar las máquinas, el servicio debe disponer de el direccionamiento IP y las credenciales de acceso de las mismas. Esta información, que el usuario proporciona, puede llegar a suponer un riesgo de seguridad en el supuesto de que sea sustraída por un ciberdelincuente. Para limitar este riesgo, se da como solución limitar el uso de cualquier memoria no volátil, como los ficheros.

En el caso de que el uso de estos sea necesario, se sugiere limitar los permisos de lectura y escritura solo al usuario que vaya a ejecutar el servicio. Incluso, se recomienda limitar aún más su acceso a solo lectura, una vez el servicio se esté ejecutando y la información ya esté cargada en el servicio.

La confidencialidad de la información resultante de ejecutar el servicio sobre una máquina. Otros de los riesgos posibles a enfrentar es la fuga de datos sensibles procedentes del análisis de las máquinas, la cual amenaza integridad de las mismas. La información que se considera sensible incluye el direccionamiento IP de las máquinas, los nombres de los paquetes de los servicios de las máquinas o las versiones de los mismos, entre otros. Si esta información fuese descubierta por un atacante, éste dispondría de los datos necesarios para realizar un ataque sobre los servicios de las máquinas analizadas, pudiendo generar un incidente de seguridad. Por lo tanto, se propone no almacenar ningún dato identificativo de las



máquinas, resultantes de la ejecución. Se toma esta decisión, ya que estos datos no se consideran necesarios para desarrollar las funcionalidades del proyecto, ni aportan algún valor adicional al usuario.

La integridad y disponibilidad de las máquinas a la hora de realizar una conexión remota. El último riesgo que puede contraponer la seguridad de las máquinas partícipes del servicio es el uso de conexiones remotas. Esta operatividad, aun siendo fundamental para el análisis de máquinas remotas, manifiesta vulnerabilidades de las cuales los ciberdelicuentes pueden explotar. En el caso de que la conexión no sea segura, un atacante podría obtener información de la transferencia de datos entre máquinas, generando una vulnerabilidad que afectaría a la integridad y la confidencialidad de las mismas. Es por ello, que se recomienda usar un protocolo de conexión seguro, el cual permita el cifrado de información, extremo a extremo, por medio de un canal cifrado, como el protocolo SSH - Secure Shell (Esteban, 2014). Este, aparte cumplir con los requisitos mencionados, cuenta con módulos prediseñados para su uso en la mayoría de los lenguajes de programación, por lo que no sería necesario construir una solución.

Capítulo 4

Diseño e implementación

4.1 Arquitectura del sistema

Tras haber realizado el análisis (Capítulo 3.2 – Análisis de herramientas similares) y la especificación de los requerimientos necesarios para el desarrollo del proyecto (Capítulo 3.4 – Especificación de requisitos), se ha elaborado la siguiente arquitectura (Ilustración 4.1.1), dividida en cuatro capas, conteniendo cada una de ellas su propia lógica y funciones definidas.

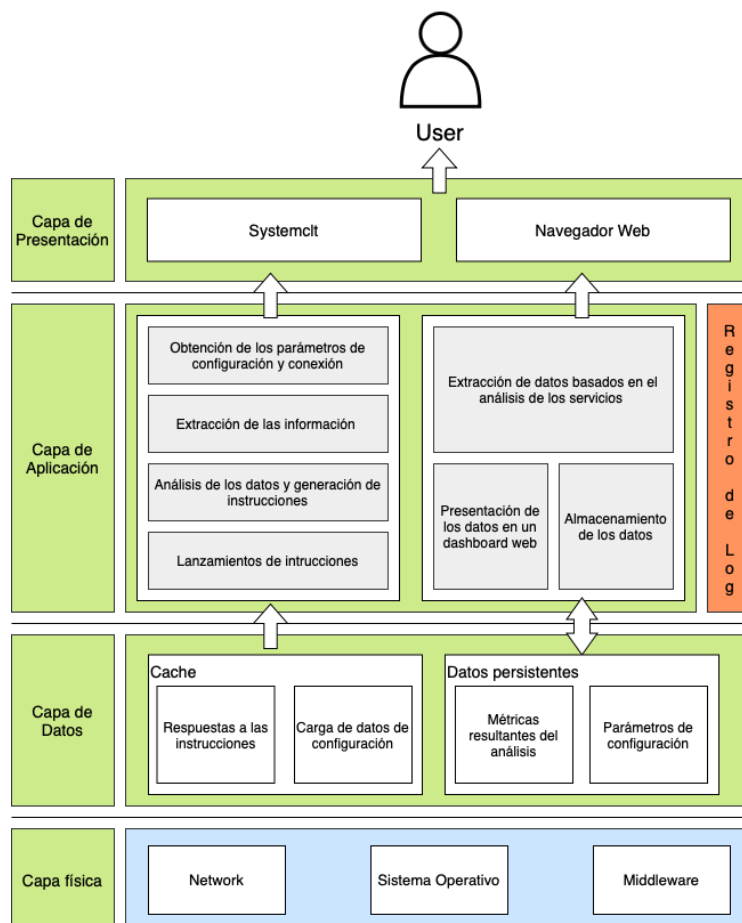


Ilustración 4.1.1 - Arquitectura del sistema



Estas cuatro capas han sido desarrolladas de manera que cada una de ellas se encuentra encapsulada, siendo únicamente necesaria la comunicación con las capas adyacentes a cada una de estas. Las capas implementadas son las siguientes:

Capa física o de infraestructura. Se trata de la capa más baja de la arquitectura. En ella se encuentran todos los componentes físicos necesarios para la ejecución del servicio. En el caso del servicio desarrollado se incluyen:

- Una máquina, donde el servicio se encuentra alojado y cuya infraestructura garantiza el almacenamiento, la disponibilidad y el rendimiento del servicio cuando éste es ejecutado.

Este servidor no requiere que esta máquina posea uno de los 3 sistemas operativos disponibles para analizar, aunque si debe disponer de la herramienta systemctl, el lenguaje Python 3 y sus módulos complementarios (Requisito Funcional, RF1).

- Una o más máquinas, las cuales serán analizadas y remediadas por el servicio. Para que el servicio pueda obtener la información de los servicios de cada máquina y pueda generar la remediación, estas máquinas deben contar con un sistema operativo derivado de Fedora, SUSE o Debian (Requisito Funcional, RF2) y disponer del servidor SSH activo (Requisito Funcional, RF3).

Aunque las máquinas no cumplan este requisito, el servicio podrá habilitarse sin que este de error (Requisito funcional, RF12), pero no realizará ninguna de sus operativas de análisis y remediación.

- Una infraestructura de redes, que permita la conexión con el resto de los servidores a analizar. Esta infraestructura es la encargada de transportar la información de forma segura, protegiendo la integridad de los datos que son procesados y almacenados.

Capa de datos. En esta capa se gestionan los datos involucrados con el servicio, incluso cuando el servicio no está activo. Existen dos tipos de datos que son partícipes en la ejecución de servicio:

- Datos temporales, en donde se localiza toda la información que se extrae para conocer la distribución de las máquinas y la situación de los paquetes de los servicios instalados en estas. Toda esta información es necesaria para que el servicio pueda construir y ejecutar correctamente las instrucciones que son enviadas a las máquinas para actualizar los servicios. Aun así, esta información no es almacenada de forma permanente, ya que no se considera relevante ni para la continuidad del servicio ni para el usuario final.
- Datos persistentes, en los que se almacenan las métricas resultantes de la ejecución. Estos datos, cuando el servicio este activo, servirán para alimentar al dashboard interactivo, con el cual el usuario puede obtener una situación concreta del estado de los paquetes de los servicios de sus máquinas. Cada vez que se reinicie el servicio, estos datos no serán mostrados en el dashboard, pero si se mantendrán en un nuevo documento en el directorio /hermesd/.

Capa de aplicación. Capa principal al ser donde se encuentran toda la operativa del servicio. En esta se gestiona el análisis de servicio en tres procesos diferenciados:

1. Conexión y obtención de datos mediante túnel seguro. Para poder conectar las máquinas analizadas con la máquina analizadora se establece, con el módulo de Python 3 Paramiko (Forcier, 2022), un túnel seguro mediante SSH (Requisito funcional, RF3). Creando así una conexión cifrada entre los sistemas involucrados. Esta conexión permite a la máquina analizadora obtener los datos necesarios para el análisis del servicio, asegurando la confidencialidad y la integridad de la información transmitida.
2. Análisis de versiones y remediación. Una vez establecida la conexión segura entre las máquinas, se lleva a cabo el análisis de los servicios instalados. Este proceso esta dividido en 3 partes:



1. Primero, el servicio **obtiene el nombre de los paquetes de los servicios que la máquina tiene y sus versiones instaladas**, almacenando esta información de manera temporal (Requisito funcional, RF4).
2. Tras haber obtenido esta información, y para cada servicio, se obtiene la última versión disponible, y **se compara con versión instalada, evaluando si estas son retrocompatibles**. Si ambas versiones son retrocompatibles, el servicio registra el su nombre en una lista almacenada en la memoria temporal (Requisito funcional, RF5). En caso contrario, se omite y se continua con el siguiente (Requisito funcional, RF6).
3. Una vez obtenido el listado de servicios retrocompatibles, se **genera una instrucción**, que es enviada por SSH a la máquina para actualizar la versión de los paquetes de los servicios. Esta instrucción varía dependiendo del sistema operativo de cada máquina, ya que estos cuentan con distintos gestores de paquetes (Requisito funcional, RF7). Tras actualizar todos los paquetes, **se reinician todos los servicios que no afecten a la disponibilidad de la máquina** para que funcionen con la nueva versión instalada.

Este proceso se realiza sobre todas las máquinas identificadas en el fichero de configuración de manera secuencial. Cuando todas las máquinas hayan sido analizadas, el servicio pasa a un estado de reposo el cual durará el tiempo que el usuario especifique en el fichero de configuración (Requisito funcional, RF10). El tiempo predefinido es de 6 horas.

3. Generación de datos y creación de un dashboard interactivo. Además de analizar y remediar los servicios, en esta capa se construyen distintas métricas las cuales alimentan el dashboard interactivo implementado.

Capa de presentación. En esta capa se encuentran todos los componentes que interactúan con el usuario, siendo en algunos necesaria su intervención para que el servicio pueda funcionar. Estos componentes son:

- Fichero de configuración. En este, el usuario podrá incluir y/o modificar diferentes parámetros previamente a la ejecución de servicio, en los que se incluyen las IPs de las máquinas a analizar, el tiempo de inactividad entre ventanas de escaneos y credenciales de acceso a las máquinas que analizar (Requisito no funcional, RNF1).

Aunque el fichero sea modificado, los parámetros de configuración no podrán ser alterados cuando el servicio se encuentre activo. Los cambios en la configuración del servicio solo podrán ser alterados cuando este sea reiniciado.

- Comandos de instalación, desinstalación y ejecución. Para que el usuario pueda instalar, desinstalar y ejecutar el servicio el usuario, se requieren 2 comandos los cuales deben ser ejecutados a través de una terminal UNIX. Estos comandos son los siguientes:

Comando	Función
sh	Ejecución de los scripts de instalación y desinstalación automática de los ficheros de configuración y scripts del servicio (Requisitos funcionales, RF13 y RF14)
systemctl	<ul style="list-style-type: none"> • Comprobar instalación del servicio • Iniciar o parar el servicio • Habilitación del servicio al arranque de la máquina

Tabla 1 - Comandos necesarios para el servicio y su función



En el proceso de instalación, los ficheros y scripts del servicio serán instalados exclusivamente con permisos de lectura y escritura (solo si es necesario) para el usuario que ejecute el fichero de instalación (Requisito no funcional, RNF2 y RNF3). Ningún otro usuario y/o grupo tendrá acceso a estos, por lo que este usuario deben tener permisos suficientes para lanzar el servicio.

- Dashboard interactivo. Mediante cualquier navegador web, el usuario tendrá la capacidad de obtener información relativa a las máquinas analizadas y el estado de sus servicios instalados a través del dashboard generado (Requisito funcional, RF16).

El dashboard únicamente estará disponible cuando el servicio este activo y se podrá observar desde la máquina encargada de realizar el análisis (Requisito funcional, RF17). Este también podrá ser visto desde cualquier otra máquina si así se configura. Aunque, no se aconseja al vulnerar la integridad y trazabilidad de los datos.

4.2 Modelo de clases de diseño

Siguiendo como base el diagrama de clases realizado durante el análisis del proyecto (Ilustración 3.5.2) y, tras haber realizado un estudio de las herramientas disponibles, se ha decidido que la mejor solución para estructurar el proyecto es usar un modelo de módulo/scripts. Se ha decidido usar este modelo debido a que este nos permite implementar el servicio más ágil y fácilmente. Esto se debe a que el modelo nos brinda una estructura de scripts viable para la creación de ficheros reconocidos como un servicio y para la programación de los instaladores y desinstaladores. El detalle de todos los ficheros es:

- **Instalador y desinstalador**: fichero bash con las instrucciones necesarias para instalar y/o desinstalar los módulos necesarios y copiar/eliminar los ficheros a la estructura de directorios del sistema.

- **Fichero de configuración:** fichero tipo `'config'`, al cual el usuario debe modificar para establecer los parámetros de configuración del servicio.
- **Ficheros de configuración de servicio:** dos ficheros de tipo `'service'`, con el cual el sistema reconoce los scripts como un servicio. Se encuentra dividido en dos ficheros, uno para el servicio y otro para el dashboard, debido a errores de concurrencia entre ambos. Aunque, no será necesario ejecutar el servicio para el dashboard para levantarlo.
- **Fichero de almacenamiento de los resultados:** fichero tipo CSV, donde el servicio almacena las métricas resultantes de la ejecución. Aparte, se generan más documentos, si el servicio es parado, con la información de la última ejecución.
- **Script de configuración del servicio:** ficheros Python 3 donde se almacena toda la lógica del servicio y del dashboard.

En la siguiente ilustración (Ilustración 4.2.1) se muestran, en mayor detalle, los métodos, funciones y bibliotecas necesarias de todos los ficheros del servicio.

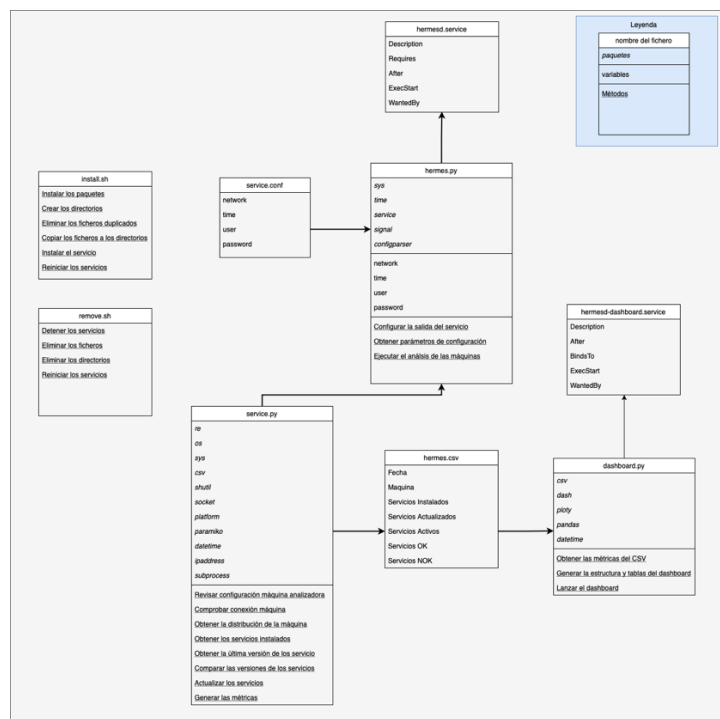


Ilustración 4.2.1 - Modelo de clases de diseño



4.3 Diseño físico de datos

El diseño de los datos que se ha implementado en el servicio afecta tanto a la configuración del servicio como a los datos resultantes de la ejecución. Estos datos cuentan con sistemas de almacenamiento distintos, los cuales almacenan estructuras de datos, valores y tipos de datos diferentes.

Para la configuración del servicio se registran cuatro valores, los cuales deben de ser cumplimentados por el usuario. Los valores se encuentran almacenados en fichero de configuración denominado **service.conf**, localizado en el directorio /etc/hermesd/. Los valores de este fichero son:

Nombre	Descripción	Tipo	Valores permitidos
network	Direccionamiento IP de las máquinas a analizar (Requisito funcional, RF11)	String	<ul style="list-style-type: none">• Dirección IP• Direcciones IP• Rango de IPs• Red
user	Usuario de acceso a las máquinas a analizar. Este usuario debe de ser único para todas las máquinas	String	Nombre del usuario
password	Contraseña del usuario de acceso	String	Contraseña para el usuario

time	Tiempo de espera entre análisis en horas	Integer	[1 – N]
------	--	---------	---------

Tabla 2 - Descripción de los datos necesarios para la ejecución

Los valores plasmados en este fichero serán luego extraídos, en el momento del arranque del servicio, para configurar el análisis en base a las necesidades del usuario. Esta extracción se realiza mediante el módulo de Python ConfigParser (PyPI, 2021).

En cambio, los datos generados a partir de la ejecución del servicio almacenan diferentes métricas que brindan una imagen detallada del estado de los servicios del sistema. Estas métricas son almacenadas en el documento CSV **hermes.csv**, localizado en el directorio */hermesd/* (Requisito funcional, RF15). Cuando se deshabilita el servicio, los datos en este documento son transferidos a uno nuevo con la fecha de deshabilitación del servicio como nombre (Requisito funcional, RF19 y RF20). Los valores que se registran en este documento son:

Nombre	Descripción	Tipo	Valores permitidos
Fecha	Fecha y hora de la ejecución de la máquina.	String	YYYY-MM-DD HH:MM:SS
Sistema Operativo	Sistema operativo de la máquina.	String	<ul style="list-style-type: none"> Fedora Debian SUSE



Servicios Instalados	Número de paquetes de servicios que se encuentran instalados	Integer	[1 – N]
Servicios Actualizados	Número de paquetes de servicios que han sido actualizados durante el análisis	Integer	[0 – N]
Servicios OK	Número de paquetes de servicios que disponen de la última versión	Integer	[1 – N]
Servicios NOK	Número de paquetes de servicios que no disponen de la última versión disponible instalada	Integer	[0 – N]

Tabla 3 - Descripción de los datos resultantes de la ejecución

4.4 Diseño de la interfaz de usuario

Al ser el producto desarrollado un servicio de Linux, este no cuenta con una interfaz de usuario que permita ejecutar sus funciones disponibles. Este es ejecutado a través de una terminal UNIX gracias a la herramienta de administración de servicios systemctl. El usuario solo dispondrá de la habilitar o deshabilitar el servicio, en el caso de que se quieran usar o no sus operativas. Aun así, desde una terminal, se pueden observar diferentes datos de la ejecución del servicio como el estado del servicio o el log de la herramienta, gracias a diferentes comandos como systemctl status o journalctl -u.

A pesar no contar con una interfaz, sí que se dispone de un dashboard interactivo creado mediante el módulo Dash (Plotly, 2019). A través de este, el usuario puede observar, mediante gráficas, diferentes métricas que se obtienen a partir de los datos almacenados de la ejecución del servicio.

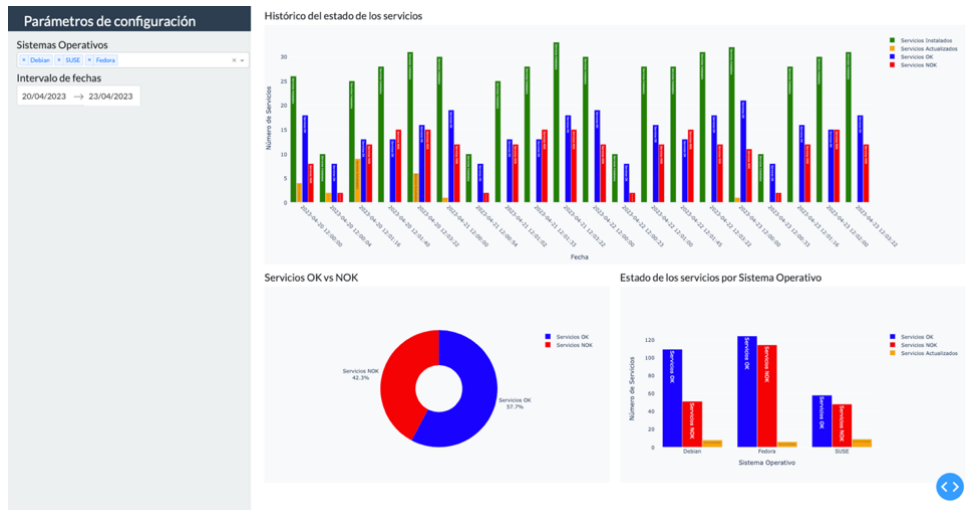


Ilustración 4.4.1 - Página principal dashboard

Adicionalmente, y dado que se trata de un dashboard interactivo, se han incorporado ciertos parámetros que posibilitan una visualización más precisa de la información (Requisito funcional, RF18). El usuario podrá obtener el detalle de



tanto la situación de cada uno de los sistemas operativos analizados (Ilustración 4.4.2) como de todos ellos en conjunto (Ilustración 4.4.1). Además de poder conocer la situación de los paquetes de los servicios, dentro un periodo de tiempo concreto definido por el usuario (Ilustración 4.4.3)

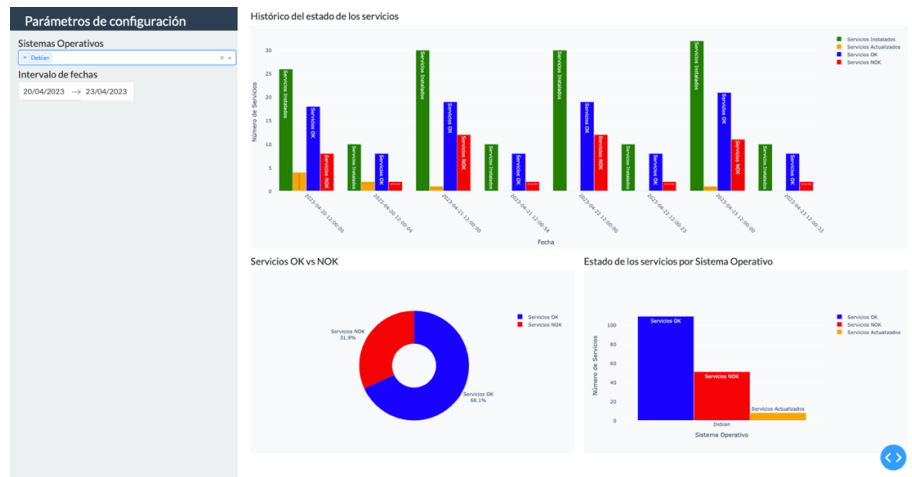


Ilustración 4.4.2 - Datos de las máquinas con S.O. derivado de Debian

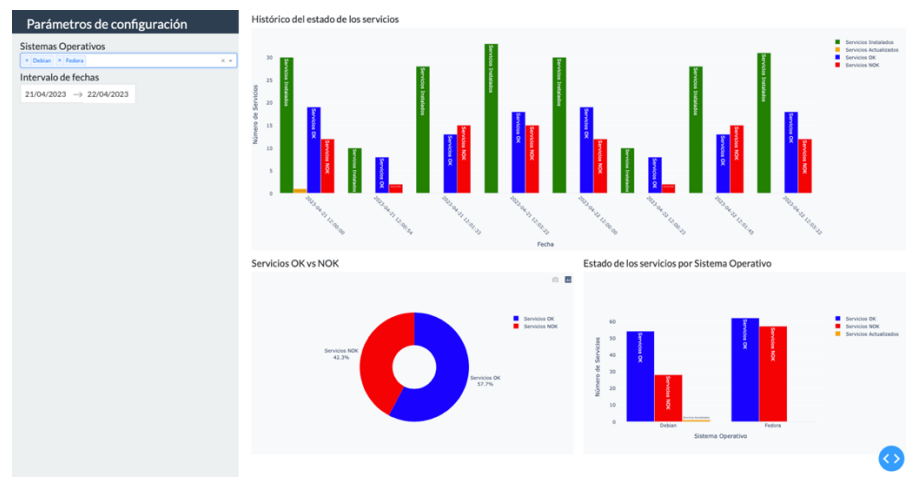


Ilustración 4.4.3 - Datos de máquinas Debian y Fedora en periodo específico

El dashboard se encuentra soportado sobre una interfaz web y podrá ser accesible desde cualquier máquina, siempre que el firewall esta configurado adecuadamente y el servicio esté habilitado.

4.5 Entorno de construcción

A lo largo de la ejecución de este proyecto se han usado distintas herramientas que han facilitado el desarrollo y las pruebas del servicio. Todas las herramientas que se han usado se encuentran catalogadas como software libre, es decir su uso no ha requerido de un pago. Las herramientas que se han usado son las siguientes

- **Terminal UNIX.** Al haber usado para el desarrollo del servicio dos máquinas con sistema operativos MacOS y Debian, se ha hecho uso de sus terminales UNIX preinstaladas. Estas se han usado para la ejecución del servicio en todo el proceso de desarrollo, para la ejecución de las pruebas una vez se había finalizado el desarrollo y para el control de las diferentes versiones a través de la herramienta Git.
- **Visual Studio Code.** Empleada para el desarrollo de los scripts de Python 3 y los ficheros de configuración necesarios para el servicio. Se decidió utilizar esta herramienta, en vez de otro editor de texto de terminal, debido a las facilidades que ofrece para programar gracias a los módulos que permiten el autocompletado de código.
- **Git.** Herramienta cuyo principal propósito es el control de las versiones del código con el fin de almacenar un registro de todo el trabajo realizado durante la realización del proyecto.
- **GitHub.** Servicio web que facilita el almacenamiento de las versiones registradas a través de git en un repositorio en línea.
- **AWS.** Herramienta web, proporcionada por Amazon, en la que se han ejecutado varias instancias de los diferentes sistemas operativos que cubren el servicio. Al contrario que el resto de las herramientas, sí que se ha requerido de un pago para su uso.



4.6 Plan de pruebas

Para validar la implementación del servicio, se ha diseñado un plan de pruebas dividido en dos partes. Este plan ha sido desarrollado con el objetivo de validar tanto los requisitos establecidos al comienzo del proyecto, como también las respuestas de los métodos implementados.

En la primera parte de las pruebas, se ha generado una batería de pruebas unitarias que asegurarán que las respuestas dadas por los métodos sean iguales a los valores esperados. Esta batería estudiará cada casuística que se pueda dar en la conexión a las máquinas, la extracción de la información, el análisis de los datos, la generación de instrucciones y la elaboración de métricas, incluyendo los casos en donde se obtengan errores. Tanto la conexión SSH, los registros de los parámetros de configuración y el almacenamiento de información no deberán ser verificados al usar módulos externos ya testeados. Las pruebas planteadas cubren todos los sistemas operativos que se encuentran dentro del alcance del proyecto y cada uno de los casos que se debe de contar los test mínimos necesarios para cerciorase de que las pruebas sean válidas.

Todos los test unitarios se encuentran implementados en varios ficheros de Python, incluidos en la carpeta [/tfg/apt/tests](#), pudiendo ser ejecutados si estos son previamente configurados. Estos tienen como función validar de una forma tangible la implementación correcta de gran parte de los requisitos planteados. Estas pruebas son:

ID Req.	ID Test	Nombre	Resultado
RF1	TS01	Test comprobación servicio en máquina sin Python 3	OK. Se comprueba que el servicio da error al no disponer del intérprete de Python 3

RF1	TS02	Test lanzamiento servicio en máquina con Python 3	OK. Se comprueba que el servicio no da error alguno
RF1	TS03	Test comprobación máquina analizadora sin la herramienta systemctl	OK. Se comprueba que el servicio da FALSO si no se dispone de la herramienta Systemd
RF1	TS04	Test comprobación máquina analizadora con la herramienta systemctl	OK. Se comprueba que el servicio da VERDADERO y no da error
RF1	TS05	Test comprobación máquina analizadora no UNIX	OK. Se comprueba que el servicio da FALSO si la máquina no es UNIX
RF1	TS06	Test comprobación máquina analizadora UNIX	OK. Se comprueba que el servicio da VERDADERO y no da error
RF2	TS07	Test comprobación conectividad en máquina no alcanzable	OK. OK. Se comprueba, mediante paquetes ICMP, que una máquina al probar la conexión da FALSO
RF2	TS08	Test comprobación conectividad en máquina alcanzable	OK. Se comprueba, mediante paquetes ICMP, que una máquina al probar la conexión da VERDADERO y no da error
RF2	TS09	Test comprobación sistema operativo máquina fuera del alcance	OK. Se comprueba que si la máquina no esta dentro del alcance, el servicio da FALSO
RF2	TS10	Test comprobación sistema operativo máquina dentro del alcance	OK. Se comprueba que si la máquina esta dentro del alcance, el servicio da VERDADERO y no da error



RF3	TS11	Test comprobación error puerto	OK. Se comprueba que se da FALSO si no se introduce un puerto correcto
RF3	TS12	Test comprobación puerto SMB cerrado en la máquina analizada	OK. Se comprueba que no se llega al puerto 445 SMB (cerrado por defecto) de la máquina analizada, por lo que da FALSO
RF3	TS13	Test comprobación puertos SSH y HTTP abiertos en la máquina analizada	OK. Se comprueba que se llega a los puertos 22 SSH y 80 HTTP de la máquina a analizar, por lo que da VERDADERO
RF4	TS14	Test obtención de paquetes de servicios preinstalados	OK. Al lanzar el escaneo y buscar los paquetes de los servicios SSH y TAR, se obtienen los servicios
RF4	TS15	Test obtención de paquetes de servicios no instalados	OK. Al lanzar el escaneo y buscar los paquetes de los servicios RSH y RLOGIN, no se obtienen los servicios
RF5	TS16	Test acción servicio vacío	OK. Se comprueba que si el servicio analizado este vacío, no se realiza nada
RF5	TS17	Test acción servicio con versión actualizada	OK. Se comprueba que si el servicio analizado esta actualizado, no se realiza nada
RF5	TS18	Test acción servicio con versión desactualizada y retrocompatible	OK. Se comprueba que, si el servicio analizado esta desactualizado y su última versión es retrocompatible, se actualiza

RF6	TS19	Test acción servicio con versión desactualizada y no retrocompatible	OK. Se comprueba que, si el servicio analizado esta desactualizado y su última versión no es retrocompatible, no se hace nada
RF7	TS20	Test obtención instrucciones para máquinas fuera del alcance	OK. Se comprueba que no se obtienen instrucciones
RF7	TS21	Test obtención instrucciones para máquinas dentro del alcance	OK. Se comprueba que se obtienen las instrucciones para el sistema operativo
RF11	TS22	Test obtención máquina sin IP indicada	OK. Se comprueba que no se obtiene ninguna IP
RF11	TS23	Test obtención máquina de una IP	OK. Se comprueba que se obtiene una lista con la IP
RF11	TS24	Test obtención máquinas de una lista de IPs	OK. Se comprueba que se obtiene una lista con las IPs
RF11	TS25	Test obtención máquinas de un rango de IPs	OK. Se comprueba que se obtiene una lista con el rango de IPs
RF11	TS26	Test obtención máquinas de una red IP	OK. Se comprueba que se obtiene una lista con las IPs de la red
RF13	TS27	Test instalación ficheros	OK. Se comprueba que los ficheros del servicio se encuentran en la máquina tras ejecutar el script



RF14	TS28	Test desinstalación ficheros	OK. Se comprueba que los ficheros del servicio no se encuentran en la máquina tras ejecutar el script
RF16	TS29	Test instanciación dashboard	OK. Se comprueba que el dashboard se inicia al habilitar el servicio
RF17	TS30	Test dashboard cuando el servicio está deshabilitado	OK. Se comprueba que el dashboard no se instancia si el servicio está deshabilitado
RF19	TS31	Test comprobación creación del fichero de almacenamiento tras reinicio	OK. Se comprueba que el formato del nombre es correcto y cuenta con los datos
RF20	TS32	Test comprobación fichero original vacío tras reinicio	OK. Se comprueba que el fichero está vacío
RNF2	TS33	Test permisos ficheros de configuración	OK. Se comprueba que el usuario solo tiene permisos de lectura y escritura
RNF3	TS34	Test permisos scripts	OK. Se comprueba que el usuario solo tiene permisos de ejecución

Tabla 4 - Listado de test unitarios implementados

Una vez ejecutados los test y validado que se obtiene el valor esperado para cada uno de ellos sin error alguno, se ejecuta la segunda parte de las pruebas, un entorno de prueba el cual comprobará el funcionamiento completo del servicio. Esta parte de los test verificará, de forma visual, la correcta implementación de resto de los requisitos no testeados en la parte anterior.

Para esta prueba, se harán uso de 4 máquinas, 3 sistemas para realizar el análisis de sus servicios; y 1 máquina para ejecutar el servicio. Las máquinas por analizar contarán con un sistema operativo distinto, de los incluidos en el alcance del proyecto. A estas máquinas, las cuales tendrán todos sus servicios actualizados, se les instalarán 6 servicios después de iniciar el servicio. Estas máquinas contendrán todas las posibles casuísticas que se puedan dar. De estos servicios, 2 se encontrarán actualizados a la última versión disponible; 2 no estarán actualizados, pero si podrán ser actualizados a la última versión al tener versiones retrocompatibles; y por último 2 no estarán actualizados y no se podrán actualizar al no disponer de versiones retrocompatibles.

Para cerciorarse de que la prueba de entorno es un éxito, las versiones de los dos primeros servicios no deben ser modificadas, las versiones de los dos segundos deben de ser actualizadas y las versiones de los dos últimos deben conservarse. Se presenta el escenario mediante la siguiente ilustración (Ilustración 4.6.1).

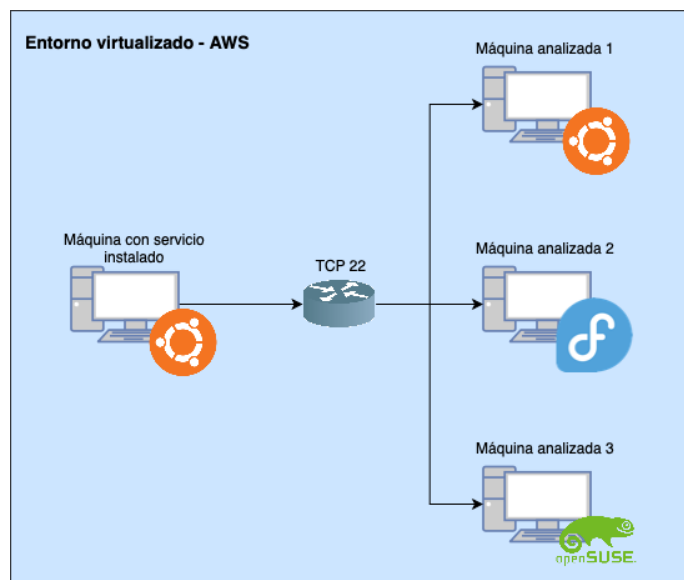


Ilustración 4.6.1 - Escenario para prueba de servicio



Tras la configuración del escenario, y la puesta en marcha del servicio, se puede comprobar, tanto revisando el dashboard generado por el servicio como los datos de los que se nutre el mismo, para comprar que las hipótesis planteadas se cumplen y los servicios se actualizan a su última versión retrocompatible. Se evidencia este supuesto mediante las siguientes ilustraciones del dashboard generado (Ilustración 4.6.2) y de los datos obtenidos (Ilustración 4.6.3).

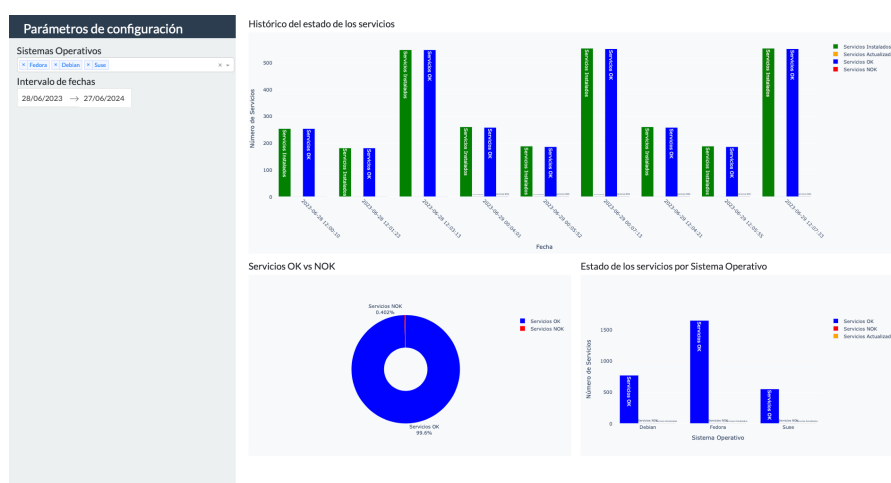


Ilustración 4.6.2 - Dashboard ejecución prueba escenario

```
Fecha,Maquina,ServiciosInstalados,ServiciosActualizados,ServiciosOK,ServiciosNOK
2023-06-28 12:00:10,Debian,254,0,254,0
2023-06-28 12:01:23,Suse,182,0,182,0
2023-06-28 12:03:13,Fedora,547,0,547,0
2023-06-29 00:04:01,Debian,260,2,258,2
2023-06-29 00:05:52,Suse,188,2,186,2
2023-06-29 00:07:13,Fedora,553,2,551,2
2023-06-29 12:04:21,Debian,260,0,258,2
2023-06-29 12:05:55,Suse,188,0,186,2
2023-06-29 12:07:33,Fedora,553,0,551,2
```

Ilustración 4.6.3 - Datos brutos ejecución prueba modelo

Tras comprobar que la solución desarrollada satisface todas las pruebas implementadas en ambas partes, se considerarán validados los requisitos establecidos y los métodos desarrollados.

4.7 Diagrama de infraestructuras de nivel 3

En el diagrama (Ilustración 4.7.1) se muestra una representación de la infraestructura de nivel tres de red, según el modelo OSI. En este se detallan los diferentes componentes y las conexiones que conforman una implementación viable del servicio que se ha desarrollado.

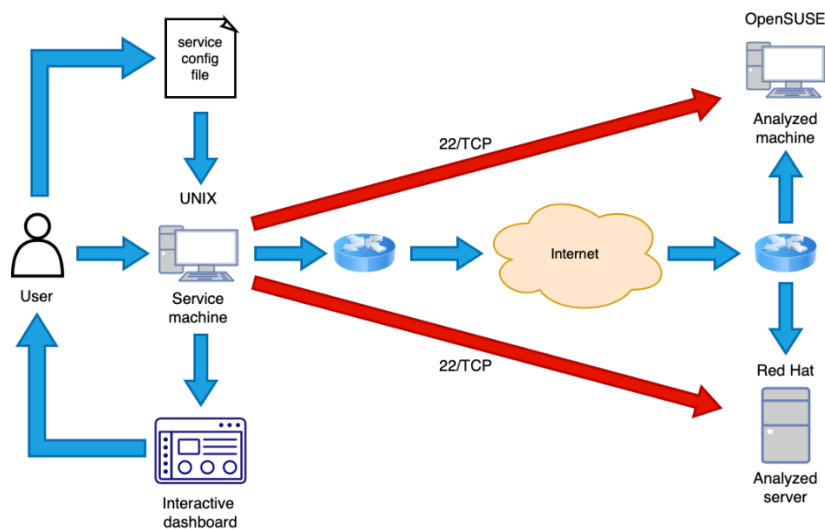


Ilustración 4.7.1 - Diagrama infraestructura nivel 3

Como se muestra, el servicio requiere de uno o más routers para que se pueda realizar la conexión TCP 22 o SSH entre la máquina analizadora y las máquinas analizadas. Estos routers deben de estar tanto conectados con las máquinas como con internet para posibilitar la conexión. En la situación, en la que no sea necesario el envío de paquetes mediante internet, se podrá hacer uso de switches en vez de routers para la comunicación entre las máquinas.

Esta ilustración no contempla todas las posibles variaciones de escenarios que se pueden dar debido al elevado número de casuísticas posibles. Es por ello que el objetivo de esta ilustración es definir los actores necesarios para el servicio, al nivel de detalle deseado.

Capítulo 5

Construcción

5.1 Referencia al repositorio de software

El software desarrollado para resolución del proyecto se encuentra almacenado en el siguiente repositorio de GitHub (Martínez Sánchez, Repositorio Trabajo Fin de Grado, 2022).

5.2 Manuales

5.2.1 Módulos necesarios

Para ejecutar este servicio, se requieren la instalación de varios módulos, no preinstalados para Python 3. Estos módulos son:

- Paramito 3.2.0, para la ejecución de conexiones SSH entre máquinas.
- Dash 2.10.2, para la creación del dashboard interactivo.
- Dash_bootstrap_components 1.4.1, para la creación del dashboard interactivo.
- Dash_core_components 2.0.0, para la creación del dashboard interactivo.
- Dash_html_components 2.0.0, para la creación del dashboard interactivo.
- Plotly 5.15.0, para la creación de gráficas dentro del dashboard.
- Pandas 2.0.2, para la manipulación y análisis de datos.

No es necesaria la instalación individual de estos módulos por parte del usuario, el fichero de instalación del servicio ya cuenta con las instrucciones necesarias para instalar los módulos en su versión necesaria.

5.2.2 Instalación del servicio

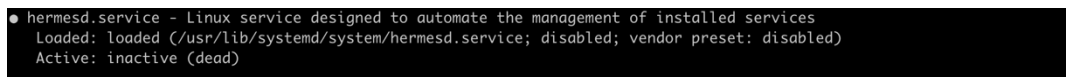
Para instalar el servicio, solo se **debe ejecutar el fichero install.sh** a través de una terminal desde el directorio tfg, mediante la siguiente instrucción:

```
sh ./install.sh
```

Una vez se haya ejecutado el fichero, se puede comprobar que el servicio ha sido **instalado correctamente** mediante la instrucción:

```
systemctl status hermesd
```

El resultado de esta ejecución debe ser igual al presentado en la siguiente ilustración (Ilustración 5.2.1).



```
● hermesd.service - Linux service designed to automate the management of installed services
   Loaded: loaded (/usr/lib/systemd/system/hermesd.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
```

Ilustración 5.2.1 - Resultado instalación servicio

5.2.3 Configuración del servicio y ejecución

Antes de ejecutar el servicio, se deberá configurar el fichero **/etc/hermesd/service.conf**. En este encontraremos los siguientes campos:

- Network. Dirección o direcciones IP de las máquinas a analizar. Este campo permite introducir:
 1. IP únicas - 192.168.56.1
 2. Varias IPs - 192.168.56.1, 192.168.56.2, 192.168.56.3
 3. Rangos de IPs - 192.168.56.110-192.168.56.114
 4. Redes completas -192.168.56.0/24
- Time. Tiempo de espera entre cada análisis, en horas.
- User. Nombre de usuario único de acceso a todas las maquinas analizadas.
- Password. Contraseña para el usuario.

Una vez se haya configurado el fichero **service.conf**, el servicio deberá ser ejecutado mediante la instrucción:

```
systemctl start hermesd
```

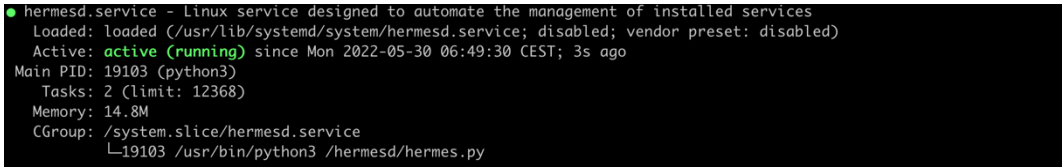
En el caso de que se quiera **ejecutar el servicio en el momento de inicio de la máquina**, se deberá usar la siguiente instrucción:

```
systemctl enable hermesd
```

Una vez ejecutada la instrucción, se puede comprobar si el servicio se ha **levantado correctamente** mediante el comando previamente usado

```
systemctl status hermesd
```

El resultado de esta ejecución dese ser igual al presentado en la Ilustración 5.2.2.



```
● hermesd.service - Linux service designed to automate the management of installed services
   Loaded: loaded (/usr/lib/systemd/system/hermesd.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2022-05-30 06:49:30 CEST; 3s ago
     Main PID: 19103 (python3)
        Tasks: 2 (limit: 12368)
       Memory: 14.8M
      CGroup: /system.slice/hermesd.service
              └─19103 /usr/bin/python3 /hermesd/hermes.py
```

Ilustración 5.2.2 - Levantamiento del servicio

5.2.4 Desinstalación del servicio

Al igual que la instalación del servicio, para desinstalarlo **solo se debe ejecutar el fichero remove.sh** desde una terminal de la siguiente forma:

```
sh ./remove.sh
```

5.2.5 Acceso al dashboard

Para que el usuario pueda acceder al dashboard con los datos del servicio, desde cualquier navegador instalado en la máquina, se debe acceder la siguiente dirección:

<http://<IP Máquina Analizadora>:8020>

5.2.6 Ejecución de test unitarios

Para ejecutar los tests, primero se deberán configurar las credenciales de acceso de la máquina, en los ficheros incluidos dentro del directorio *app/test/*. Estas credenciales son imprescindibles para la ejecución de las pruebas, y deben de contar con privilegios de superusuario en la máquina.

Una vez configuradas las credenciales en todos los ficheros, desde el directorio *app* y con un usuario con privilegios de administrador, se podrán ejecutar los test mediante la instrucción:

```
make test
```

Aparte, toda esta información se encuentra almacenada en el mismo repositorio que el código, concretamente en el fichero README (Martínez Sánchez, README Trabajo Fin de Grado, 2022).

Capítulo 6

Conclusiones y líneas futuras

El objetivo principal de este trabajo de fin de grado fue desarrollar un servicio UNIX, con el que poder analizar y actualizar los servicios instalados de una o más máquinas con distribuciones específicas de Linux. Gracias a la implementación de este servicio automatizado, se buscó desinhibir al usuario final del proceso de actualizar regularmente los servicios, creando un proceso simple, ágil y asequible, con el que poder limitar el riesgo de sufrir un ciberataque.

Tras análisis del proyecto, se pudo comprobar que actualmente existen un alto número de ataques debido a la falta de mantenimiento de los equipos. Además, la ausencia de herramientas accesibles y sencillas, que brinden soluciones para automatizar la actualización de los servicios, conlleva a que este tipo de vulnerabilidades sean un punto crítico para cualquier infraestructura. Esto remarca la importancia de este proyecto ya que, al no ser un proyecto que haya conllevado una dificultad técnica alta, genera un alto impacto sobre la limitación de un gran número de posibles riesgos de seguridad.

Los resultados obtenidos en este proyecto, en especial los recogidos durante la etapa de ejecución de las pruebas, han determinado que la solución presentada cumple tanto con los requisitos detallados para este proyecto como con el objetivo principal del mismo. Demostrando de forma práctica, el papel crucial de este servicio al limitar, de manera efectiva, el riesgo de ataques y accesos no autorizados ocasionados por vulnerabilidades. Esto se debe a que, la creación de un servicio automático de actualización reduce los tiempos de parcheo de versiones de los servicios, acortando así las ventanas de los ciberdelicuentes para explotar posibles fallos.

Aun así, y tras haber finalizado el desarrollo de servicio, se han podido detectar errores o fallos, producidos durante el proceso de análisis y descubiertos tras finalizar su implementación, principalmente relacionados con la interacción usuario-servicio. Si es verdad que la operativa del servicio es simple y adecuada, la prestación del servicio llega a ser un poco complicada. La descarga del paquete, el uso de la terminal para la instalación, la necesidad de acceder a ciertos ficheros para configurar el servicio o la limitación de métricas del resultado hace que este servicio solo esté disponible para usuarios con cierto nivel en la administración de sistemas Linux. Por ello se considera que, como líneas a futuro, se estudien dos posibles mejoras que son:

1. La viabilidad de implementar un interfaz visual sencillo y amigable para el usuario final, con la que pueda instalar, configurar y ejecutar el servicio en pocos pasos. De esta forma eliminamos la necesidad de usar cualquier tipo de comando de terminal de Linux y facilitamos el uso de la herramienta para cualquier usuario que requiera usarla. Esta interfaz no solo favorecería el uso de la herramienta, sino que también reduciría las posibilidades de realizar una configuración incorrecta por parte del usuario.
2. La posibilidad de almacenar información más detallada de las máquinas y del resultado de su análisis. Aunque el dashboard web actual refleje información relevante y suficiente para que el usuario pueda obtener una imagen con la situación de los servicios en sus máquinas, es cierto que mucha información con respecto a las propias máquinas no es recogida por motivos de seguridad. Las direcciones IP, el nombre de los servicios o sus versiones son algunos de los campos que se decidieron no registrar por los riesgos que conllevaban guardar esta información. Estos datos, aun acarreando cierto riesgo, se cree que puede ser mitigados si se implementan las medias de seguridad adecuadas. Como, por ejemplo, se ha pensado en estudiar la posibilidad de sustituir los ficheros CSV por una



base de datos cifrada con la que almacenar toda la información, securizando así el almacenamiento de esta información y permitiendo el registro de los datos mencionados, entre otros.

Para concluir, y a modo de resumen, este trabajo de fin de grado ha logrado su objetivo principal al desarrollar un servicio que permita actualizar los servicios de distintas distribuciones de Linux de manera autónoma. Los resultados obtenidos demuestran un impacto positivo sobre la seguridad de ciertas infraestructuras la limitando su riesgo y facilitando su uso. Y, se espera que este proyecto sirva de base para la creación de otras herramientas asequibles, que permitan a los usuarios hacer uso de sus sistemas de la forma más segura.

Capítulo 7

Bibliografía

Alarcón, J. M. (24 de Octubre de 2020). *Configurar Ubuntu Linux Server para que se actualice de manera automática*. Recuperado el 2 de Abril de 2022, de JASoft.: <https://www.jasoft.org/Blog/post/configurar-ubuntu-linux-server-para-que-se-actualice-de-manera-automatica>

Armesto, J. (2013 de Noviembre de 2023). *¿Qué es el versionamiento semántico y por qué es importante?* Recuperado el 12 de 6 de 2022, de Jose Armesto's Blog: <https://blog.armesto.net/que-es-el-versionamiento-semantico-y-por-que-es-importante/>

AWS. (2022). *Amazon EC2*. Recuperado el 22 de Marzo de 2022, de AWS: https://aws.amazon.com/es/ec2/?nc2=h_ql_prod_fs_ec2

Bardot, Y. (2021). *Administración de un sistema Linux (2a edición)*. Editorial ENI.

Berzal, F. (Septiembre de 2005). *El ciclo de vida de un sistema de información*. Recuperado el 10 de Febrero de 2022, de Flanagan URG: <http://flanagan.ugr.es/docencia/2005-2006/2/apuntes/ciclovida.pdf>

Cybersecurity Pricing. (24 de Septiembre de 2018). *Tenable vs. Qualys Pricing and Cost Comparison*. Recuperado el 21 de Noviembre de 2021, de Cybersecurity Pricing: <https://cybersecuritypricing.org/tenable-vs-qualys-pricing-and-cost-comparison/>

Debian Foundation. (30 de Diciembre de 2021). *Unattended Upgrades*. Recuperado el 5 de Enero de 2022, de Debian Wiki: <https://wiki.debian.org/UnattendedUpgrades>

Deloitte. (Enero de 2022). *El estado de la ciberseguridad en España*. Recuperado el 5 de Enero de 2022, de Deloitte: <https://www2.deloitte.com/es/es/pages/risk/articles/estado-ciberseguridad.html>

Departamento de Defensa de los E.E.U.U. (30 de Enero de 2023). *Cybersecurity Reference Architecture*. Recuperado el 5 de Junio de 2023, de dodcio.defense.gov: <https://dodcio.defense.gov/Portals/0/Documents/Library/CS-Ref-Architecture.pdf>

Deyimar, A. (7 de Diciembre de 2022). *Cómo administrar y listar servicios en Linux*. Recuperado el 5 de Junio de 2023, de Hostinger Tutoriales: <https://www.hostinger.es/tutoriales/administrar-y-listar-servicios-en-linux>

Dubey, D. (28 de Febrero de 2022). *How to Setup Python Script Autorun As a Service in Ubuntu 18.04*. Recuperado el 13 de Marzo de 2022, de WebSoftTechs: <https://websofttechs.com/tutorials/how-to-setup-python-script-autorun-in-ubuntu-18-04/>

El Taller del BIT. (Febrero de 2020). *Crear Servicios con Systemd en Ubuntu*. Recuperado el 11 de Marzo de 2022, de El Taller del BIT: <https://eltallerdelbit.com/crear-servicio-systemd-ubuntu/>

Esteban, E. V. (Enero de 2014). *Servicios de acceso remoto II: SSH*. Recuperado el 11 de Marzo de 2022, de Informática UV: <http://informatica.uv.es/it3guia/AGR/apuntes/teoria/documentos/SSH.pdf>

Fiaño Rodríguez, J., Martínez Varela, E., Piñeiro Bermúdez, S., Santos Espido, S., & Sánchez Pardal, V. (2021). *SYSTEMD*. Recuperado el 12 de Marzo de 2022, de Universidad de La Coruña:

<https://www.dc.fi.udc.es/~afyanez/Docencia/2021/Grado/Trabajos/Systemd-VSP.pdf>

Ficher, J. (21 de Junio de 2018). *Reading and Writing CSV Files in Python*. Recuperado el 28 de Abril de 2022, de Real Python: <https://realpython.com/python-csv/>

Forcier, J. (Abril de 2022). *Paramiko - API documentation*. Recuperado el 21 de Abril de 2022, de Paramiko: <https://docs.paramiko.org/en/stable/>

Galvez, V. (20 de Enero de 2022). *Create service systemctl-systemd in python*. Recuperado el 12 de Marzo de 2022, de Towards Dev: <https://towardsdev.com/create-service-systemctl-systemd-in-python-9a0e8b5ab6ae>

Gite, V. G. (9 de Diciembre de 2021). *How to use yum command on Linux (CentOS/RHEL)*. Recuperado el 9 de Abril de 2022, de CyberCiti: <https://www.cyberciti.biz/faq/rhel-centos-fedora-linux-yum-command-howto/>

Gite, V. G. (14 de Enero de 2023). *Apt Command Examples for Ubuntu/Debian Linux*. Recuperado el 1 de Febrero de 2023, de CiberCiti: <https://www.cyberciti.biz/faq/ubuntu-lts-debian-linux-apt-command-examples/>

Gunnarhj. (5 de Julio de 2021). *AptGet/Howto - Community Help Wiki*. Recuperado el 13 de Marzo de 2022, de Ubuntu: <https://help.ubuntu.com/community/AptGet/Howto>

IBM. (10 de Enero de 2023). *¿Qué es un ciberataque?* Recuperado el 5 de Junio de 2023, de IBM: <https://www.ibm.com/es-es/topics/cyber-attack>

INCIBE. (Enero de 2022). *Balance de ciberseguridad 2022*. Recuperado el 27 de 2022 de Febrero, de Incibe:

https://www.incibe.es/sites/default/files/paginas/que-hacemos/balance_ciberseguridad_2022_incibe.pdf

INCIBE. (15 de Febrero de 2023). *Hacker vs. Ciberdelincuente*. Recuperado el 5 de Junio de 2023, de INCIBE: <https://www.incibe.es/aprendeciberseguridad/hacker-vs-ciberdelincuente>

IONOS. (11 de Marzo de 2019). *El modelo en cascada: desarrollo secuencial de software*. Recuperado el 27 de Febrero de 2022, de Ionos: <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/#:~:text=Continúa-,¿Qué%20es%20el%20modelo%20en%20cascada%3F,ejecuta%20tan%20solo%20una%20vez.>

KeepCoding. (22 de Junio de 2022). *Los 5 pilares básicos de la ciberseguridad*. Recuperado el 18 de Abril de 2023, de KeepCoding: <https://keepcoding.io/blog/pilares-basicos-de-la-ciberseguridad/>

Khan, W. (24 de Agosto de 2020). *Setup a python script as a service through systemctl/systemd*. Recuperado el 15 de Marzo de 2022, de Medium: <https://medium.com/codex/setup-a-python-script-as-a-service-through-systemctl-systemd-f0cc55a42267>

Kinger, P. A., & Logan, M. A. (23 de Agosto de 2021). *Linux Threat Report 2021 1H: Linux Threats in the Cloud and Security Recommendations*. Recuperado el 12 de Enero de 2022, de Trend Micro: <https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/linux-threat-report-2021-1h-linux-threats-in-the-cloud-and-security-recommendations>

Linuxize. (24 de Febrero de 2020). *APT Command in Linux*. Recuperado el 28 de Marzo de 2022, de Linuxize: <https://linuxize.com/post/how-to-use-apt-command/>

Mandge, H. (21 de Noviembre de 2021). *Backwards Compatibility in a Software System with Systematic Reference to Java*. Recuperado el 17 de Junio de 2023, de Geeks For Geeks: <https://www.geeksforgeeks.org/backwards-compatibility-in-a-software-system-with-systematic-reference-to-java/>

Manejando Datos. (06 de 01 de 2014). *Trabajando con ficheros de configuración en Python: ConfigParser*. Recuperado el 22 de Abril de 2022, de Manejando Datos: <https://www.manejandodatos.es/2014/01/trabajando-con-ficheros-de-configuracion-en-python-configparser/>

Martínez Sánchez, J. M. (23 de Febrero de 2022). *README Trabajo Fin de Grado*. Recuperado el 24 de Junio de 2023, de GitHub: <https://github.com/josemanuel179/tfg/blob/main/README.md>

Martínez Sánchez, J. M. (23 de Febrero de 2022). *Repositorio Trabajo Fin de Grado*. Recuperado el 24 de Junio de 2023, de GitHub: <https://github.com/josemanuel179/tfg>

Ministerio de Industria, Comercio y Turismo. (Enero de 2022). *Cifras PyME Enero 2022*. Recuperado el 15 de Mayo de 2023, de iPyME: <http://www.ipyme.org/es-ES/ApWeb/EstadisticasPYME/Documents/CifrasPYME-enero2022.pdf>

OWASP Foundation. (2021). *OWASP Top Ten*. Recuperado el 10 de Enero de 2022, de OWASP: <https://owasp.org/www-project-top-ten/>

Plotly. (15 de Agosto de 2019). *Introducing Dash*. Recuperado el 5 de Mayo de 2022, de Medium: <https://medium.com/plotly/introducing-dash-5ecf7191b503>

- PyPI. (19 de Agosto de 2021). *Python Configparser 5.3.0*. Recuperado el 21 de Abril de 2022, de PyPI: <https://pypi.org/project/configparser/>
- Python Software Foundation. (Marzo de 2022). *Python 3*. Recuperado el 19 de Marzo de 2022, de Python: <https://www.python.org/about/>
- Qualys Enterprise. (Julio de 2021). *Policy Compliance - Getting Started Guide*. Recuperado el 10 de Febrero de 2022, de Qualys: <https://www.qualys.com/docs/qualys-policy-compliance-guide.pdf>
- Raj, A. (14 de Diciembre de 2020). *ConfigParser Module - Create Configuration Files in Python*. Recuperado el 1 de Mayo de 2022, de AskPython: <https://www.askpython.com/python-modules/configparser-module>
- Red Hat Enterprise. (Diciembre de 2014). *Yum command cheat sheet*. Recuperado el Abril de 2022, de Red Hat Wiki: [://access.redhat.com/sites/default/files/attachments/rh_yum_cheatsheet_1214_jcs_print-1.pdf](https://access.redhat.com/sites/default/files/attachments/rh_yum_cheatsheet_1214_jcs_print-1.pdf)
- RedHat. (25 de Noviembre de 2021). *El concepto de CVE*. Recuperado el 5 de Junio de 2023, de RedHat: <https://www.redhat.com/es/topics/security/what-is-cve>
- Saive, R. S. (27 de Enero de 2017). *45 Zypper Commands to Manage 'Suse' Linux Package Management*. Recuperado el 2 de Abril de 2022, de TecMint: <https://www.tecmint.com/zypper-commands-to-manage-suse-linux-package-management/>
- Schroder, C. (20 de Diciembre de 2011). *Managing services on Linux with systemd*. Recuperado el 19 de Abril de 2022, de Linux.com: <https://www.linux.com/training-tutorials/managing-services-linux-systemd/>

Simic, S. F. (6 de Mayo de 2018). *How to Use the Apt-Get Command in Linux*.

Recuperado el 30 de Marzo de 2022, de Phoenixnap:

<https://phoenixnap.com/kb/how-to-use-apt-get-commands>

Systemd. (2022). *Systemd - System and Service Manager*. Recuperado el 24 de

Marzo de 2022, de Systemd: <https://systemd.io>

Tevault, D. T. (2022). *Linux Service Management Made Easy with Systemd:*

Advanced Techniques to Effectively Manage, Control, and Monitor Linux Systems and Services. Packt Publishing.

The MITRE Corporation. (Marzo de 2022). *CVE Program Mission*. Recuperado el 20

de Marzo de 2022, de CVE: <https://www.cve.org>

Tomar, A. T. (30 de Enero de 2022). *Dash for Beginners: Create Interactive Python*

Dashboards. Recuperado el 6 de Mayo de 2022, de Medium:

<https://towardsdatascience.com/dash-for-beginners-create-interactive-python-dashboards-338bfc66ffa4>

Tyler, C. (2006). *Fedora Linux: A Complete Guide to Red Hat's Community*

Distribution. O'Reilly Media, Inc.