UNIVERSIDAD SAN PABLO - CEU

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA DE SISTEMAS DE INFORMACIÓN



TRABAJO FIN DE GRADO

**Desarrollo de una herramienta Linux para actualización automática de servicios**

Autor: José Manuel Martínez Sánchez

Tutor: Álvaro Sánchez Picot

Junio 2023

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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se ha inspirado en la vulnerabilidad presentada en el informe de vulnerabilidades creado por la OWASP, Vulnerable and Outdated Components (OWASP Foundation, 2021).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 ciberincidente.

El objetivo de este proyecto es generar una solución, para usuarios con conocimientos mínimos 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, OpenSuse o Fedora.

Palabras Clave

Servicio UNIX, Systemd, Debian, OpenSUSE, 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it has been inspired by the vulnerability presented in the vulnerability report created by OWASP, Vulnerable and Outdated Components (OWASP Foundation, 2021).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, OpenSuse or Fedora.

Keywords

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

Índice de contenidos

[Capítulo 1 Introducción 1](#_Toc136896639)

[1.1 Contexto 1](#_Toc136896640)

[1.2 Objetivos del proyecto 2](#_Toc136896641)

[1.3 Contenidos de la memoria 3](#_Toc136896642)

[1.4 Terminología básica del proyecto 4](#_Toc136896643)

[Capítulo 2 Gestión del proyecto 7](#_Toc136896644)

[2.1 Modelo de ciclo de vida 7](#_Toc136896645)

[2.2 Papeles desempeñados en el proyecto 7](#_Toc136896646)

[2.3 Planificación 8](#_Toc136896647)

[2.4 Presupuesto 10](#_Toc136896648)

[2.5 Ejecución 11](#_Toc136896649)

[Capítulo 3 Análisis 13](#_Toc136896650)

[3.1 Introducción del análisis 13](#_Toc136896651)

[3.2 Análisis de herramientas similares 13](#_Toc136896652)

[3.3 Justificación de las tecnologías de uso 14](#_Toc136896653)

[3.4 Especificación de requisitos 15](#_Toc136896654)

[3.5 Análisis de los casos de uso y de las clases de análisis 19](#_Toc136896655)

[3.6 Análisis de seguridad 22](#_Toc136896656)

[Capítulo 4 Diseño e implementación 24](#_Toc136896657)

[4.1 Arquitectura del sistema 24](#_Toc136896658)

[4.2 Modelo de clases de diseño 29](#_Toc136896659)

[4.3 Diseño físico de datos 30](#_Toc136896660)

[4.4 Diseño de la interfaz de usuario 33](#_Toc136896661)

[4.5 Entorno de construcción 36](#_Toc136896662)

[4.6 Plan de pruebas 37](#_Toc136896663)

[4.7 Diagrama de infraestructuras de nivel 3 43](#_Toc136896664)

[Capítulo 5 Construcción 45](#_Toc136896665)

[5.1 Referencia al repositorio de software 45](#_Toc136896666)

[5.2 Manuales 45](#_Toc136896667)

[5.2.1 Módulos necesarios 45](#_Toc136896668)

[5.2.2 Instalación del servicio 45](#_Toc136896669)

[5.2.3 Configuración del servicio y ejecución 46](#_Toc136896670)

[5.2.4 Desinstalación del servicio 47](#_Toc136896671)

[5.2.5 Acceso al dashboard 47](#_Toc136896672)

[Capítulo 6 Conclusiones y líneas futuras 49](#_Toc136896673)

[Capítulo 7 Bibliografía 53](#_Toc136896674)

Índice de ilustraciones

[Ilustración 2.3.1 - Diagrama Gantt con la planificación Inicial del proyecto 10](#_Toc138111752)

[Ilustración 3.5.1 - Análisis casos de uso 19](#_Toc138111753)

[Ilustración 3.5.2 - Diagrama de ficheros 21](#_Toc138111754)

[Ilustración 4.1.1 - Arquitectura del sistema 24](#_Toc138111755)

[Ilustración 4.2.1 - Modelo de clases de diseño 30](#_Toc138111756)

[Ilustración 4.4.1 - Página principal dashboard 34](#_Toc138111757)

[Ilustración 4.4.2 - Datos de las máquinas con S.O. derivado de Debian 35](#_Toc138111758)

[Ilustración 4.4.3- Datos de máquinas Debian y Fedora en un periodo específico 35](#_Toc138111759)

[Ilustración 4.6.1 - Escenario para prueba de servicio 42](#_Toc138111760)

[Ilustración 4.6.2 - Dashboard ejecución prueba escenario 43](#_Toc138111761)

[Ilustración 4.6.3 - Datos brutos ejecución prueba modelo 43](#_Toc138111762)

[Ilustración 4.7.1 - Diagrama infraestructura nivel 3 44](#_Toc138111763)

[Ilustración 5.2.1 - Resultado instalación servicio 46](#_Toc138111764)

[Ilustración 5.2.2 - Levantamiento del servicio 47](#_Toc138111765)

Índice de tablas

[Tabla 1 - Comandos necesarios para el servicio y su función 28](#_Toc138111766)

[Tabla 2 - Descripción de los datos necesarios para la ejecución 31](#_Toc138111767)

[Tabla 3 - Descripción de los datos resultantes de la ejecución 33](#_Toc138111768)

[Tabla 4 - Listado de test unitarios implementados 41](#_Toc138111769)

# Introducción

## 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, de detallar sus causas y definir métodos que existen 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 que se quiere limitar con este proyecto, que son las **vulnerabilidades en los dispositivos debido a la falta de mantenimiento y actualización de sus servicios**. Concretamente en las 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, herramientas de seguridad especializadas y agentes de seguimiento de vulnerabilidades que facilitan su labor.

Sin embargo, para el resto de las empresas que requieren el uso de las tecnologías para su negocio, pero no de capital humano cualificado ni medidas de seguridad debido a su presupuesto, realizar el seguimiento de estas vulnerabilidades puede llegar a ser una tarea imposible. En este grupo de empresas se encuentran las Pymes, siendo estas el 99,8% del total de las empresas en España (Ministerio de Industria, Comercio y Turismo, 2022).

## 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 de la máquina. Aparte de proveer 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, qué 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.

## Contenidos de la memoria

A lo largo de este documento, se exponen detalladamente los procesos llevados a cabo para la realización del proyecto. Los 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 los manuales descritos para su correcta instalación y uso.

**6. Conclusiones**, reflexionando sobre la relevancia intrínseca de la materia del proyecto, identificando los posibles errores cometidos durante el desarrollo y detallando posibles líneas futuras de desarrollo para el proyecto.

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

## 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).
* **Retro compatibilidad**.
* **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).

# Gestión del proyecto

## Modelo de ciclo de vida

Para el desarrollo del servicio en el que está enfocado el proyecto, se usará 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 qué 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.

## 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.

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 mediante una reunión.

## 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.
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. La 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.

Gráfico, Escala de tiempo

Descripción generada automáticamente

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.

## 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, se reserva una cantidad de dinero para la realización de las pruebas, en el caso de que la capacidad actual de hardware disponible no permite ejecutarlas. De ser así, se hará uso de la infraestructura de 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.

Si el desarrollador del servicio no contase con los componentes, hardware o software, necesarios para la ejecución de este proyecto, se estima que el desembolso necesario sería superior a 2.000 euros para adquirir los elementos descritos previamente.

## 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.

Debido 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. Aparte, después de resolver los problemas, fue necesario repetir las pruebas del servicio con las nuevas modificaciones, con el fin de asegurar de nuevo 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. Cómo sé preciso en el presupuesto, los fondos fueron destinados al despliegue de varias máquinas virtuales con las distribuciones dentro del alcance, para realizar las pruebas.

# Análisis

## 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, se justificarán el uso de las herramientas técnicas que se consideran fundamentales para la finalización del proyecto.

## Análisis de herramientas similares

Actualmente, existen diferentes herramientas o servicios en el mercado, que cuentan con funciones para gestionar el mantenimiento de las versiones de los servicios instalados en distribuciones Linux.

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. Pero, existen dos hándicaps que dificultan la implementación y uso de esta herramienta, que son su precio y su dificultad técnica. La compra de este servicio puede suponer al usuario un desembolso de entre $295 a $1.995 anuales por máquina (Cybersecurity Pricing, 2018), dependiendo del tamaño de la empresa y el número de máquinas), para disponer de este módulo. Además del precio, de este servicio no incluye el sueldo de un ingeniero de seguridad, el cual es imprescindible para realizar la gestión la herramienta.

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 servicios de una única máquina. Siendo esta una solución muy similar a la planteada en este proyecto, se observan diversas limitaciones que comprometen su usabilidad. Estos problemas son la complejidad de configuración para el usuario, en especial si se cuenta con distintas distribuciones de UNIX; la problemática de ser un servicio local, que no puede ser replicado en otras máquinas; y la ineficiencia en la actualización, ya que no se toma en cuenta la retrocompatibilidad.

## 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 las todas distribuciones dentro del alcance del proyecto. Gracias a este administrador, el usuario podrá ejecutar y detener el servicio fácilmente cuando él lo deseé.

## Especificación de requisitos

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

**Requisitos funcionales**

#### El servicio debe poder ser ejecutado desde cualquier máquina con una distribución derivada de UNIX que disponga del intérprete de Python 3 y de la herramienta de gestión de servicios Systemd.

#### El servicio debe tener la capacidad de realizar sus funciones de conexión, análisis y actualización de servicios al menos sobre máquinas Linux cuya distribución sea derivada de Fedora, SUSE y Debian.

#### El servicio debe realizar las conexiones con las máquinas analizadas a través del servicio SSH.

#### 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 cada distribución.

#### El servicio deberá tener la capacidad de actualizar automáticamente los servicios de una máquina a su última versión disponible, si la versión instalada es retrocompatible con la versión a instalar.

#### En los casos donde los servicios no sean retrocompatibles con la última versión disponible, el servicio no deberá actualizarlos, independientemente de que si exista una actualización anterior retrocompatible con la versión instalada.

#### 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.

#### El servicio podrá ser ejecutado sobre la máquina local, donde el servicio es ejecutado, o sobre una o más máquinas remotas, si estas cuentan con los sistemas operativos disponibles y disponen de la capacidad de realizar conexiones remotas.

#### 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.

#### 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.

#### El servicio podrá realizar el análisis de las máquinas sobre una o más direcciones IP, rangos de direcciones IP o redes completas.

#### 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.

#### 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.

#### 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.

#### El servicio debe almacenar distintos datos resultantes de la ejecución del servicio. Estos datos no deben se 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.

#### 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.

#### El dashboard solo estará disponible cuando el servicio se encuentre habilitado. En caso contrario, este no podrá ser accedido.

#### 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.

#### Cada vez que el servicio sea habilitado la información previa será almacenada en un nuevo fichero cuyo nombre debe de identificar la fecha de creación del documento.

#### 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**

#### El usuario deberá tener la capacidad de introducir el direccionamiento de las máquinas mediante direcciones IP, rangos de direcciones IP o redes completas.

#### 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.

#### Los ficheros que sean parte del funcionamiento del servicio (scripts), tendrán únicamente permisos de ejecución para el usuario responsable de la ejecución.

#### 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.

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

## 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).

Diagrama

Descripción generada automáticamente

Ilustración 3.5.1 - Análisis casos de uso

**Caso Nº1 – Configuración de servicio**. Para que el sistema pueda ser ejecutado, este debe de ser previamente instalado y configurado. Para instalarlo, 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. Una vez instalado, deberá establecer los valores de configuración del análisis, situados en un fichero de configuración, para poder establecer el direccionamiento y 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, después de haber instalado y configurado la herramienta, mediante el comando "systemctl". Además, cuando el sistema esté habilitado, el usuario podrá acceder al dashboard con los datos generados por 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.

La ejecución de estos casos es independientemente del sistema operativo de la máquina donde se ejecute el servicio, siempre que este se encuentre dentro de los sistemas que alcanza el proyecto; y de las máquinas donde se realice el análisis.

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

* **Instalador** (install.sh): fichero bash cuya función es copiar los ficheros del proyecto, a la estructura de directorios del sistema, para poder ejecutar el servicio.
* **Desinstalador** (remove.sh): fichero bash 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 de Python** (.conf): fichero de configuración de los parámetros, a incluir por el usuario, necesarios para la ejecución del servicio.
* **Fichero de configuración de servicio** (.service): fichero de configuración necesario para que la aplicación desarrollada sea reconocida como un servicio por el sistema (Galvez, 2022).
* **Fichero de almacenamiento de los resultados** (.csv): fichero en donde se almacenarán las métricas extraídas a raíz de la ejecución del servicio.
* **Script Python de configuración del servicio** (service.py): script de Python donde se ejecutan los métodos desarrollados en "app.py".
* **Script Python de aplicación del servicio** (app.py): script de Python donde se instancian los métodos necesarios para el análisis de los sistemas y el lanzamiento de comandos.
* **Script Python de creación del dashboard** (dashboard.py): script de Python 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).

Diagrama

Descripción generada automáticamente

Ilustración 3.5.2 - Diagrama de ficheros

## 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 ciberdelicuente. 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. E incluso se recomienda limitar aún más su acceso, una vez el servicio se esté ejecutando y la información ya esté cargada.

**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, amenazando la integridad de las mismas. La información que se considera sensible incluye el direccionamiento IP de las máquinas, los nombres de los servicios 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 algun valor adicional.

**La integridad, confidencialidad 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 su desarrollo en el servicio, para el análisis de máquinas remotas, manifiesta vulnerabilidades de las cuales los ciberdelicuente 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.

# Diseño e implementación

## 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.

Diagrama

Descripción generada automáticamente

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 Systemd, 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, OpenSUSE 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 clientes 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 encuentran toda la información que se extrae para conocer la distribución de las máquinas y la situación 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 el servicio ni para el usuario final.
* Datos persistentes, en los que se almacenan los resultados de la ejecución. Estos datos, cuando el servicio esta activo, servirán para alimentar al dashboard interactivo, con el cual el usuario podrá obtener una situación concreta del estado 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 directorio específico del servicio.

**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 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 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 ambos servicios son retrocompatibles, el servicio registra el su nombre en una lista, que es almacenada en la memoria temporal (Requisito funcional, RF5). En caso contrario, se omite y se continua con el siguiente servicio (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 estos servicios. Esta instrucción varia dependiendo del sistema operativo de cada máquina, ya que estos cuentan con distintos gestores de paquetes (Requisito funcional, RF7).

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 en el fichero es de 5 horas.

1. 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. Sólo 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 | * 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. 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 (Requisito funcional, RF17).

## 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 tecnológicas disponibles, se ha decidido que la mejor solución para estructurar el proyecto es usar un modelo de módulos (similar a los módulos que pueden ser importados a un lenguaje), en vez de implementar un modelo de clases tradicional.

Además de los scripts que trabajan como módulos, el servicio contiene un fichero principal, encargado de ejecutar el servicio alimentándose de los métodos importados de los ficheros anteriores; y otros ficheros secundarios cuya única función es la instalación y desinstalación del servicio.

En la siguiente ilustración (Ilustración 4.2.1) se muestran, en mayor detalle, los métodos y funciones de cada uno de los ficheros del servicio, la relaciones entre los ficheros y las bibliotecas o módulos externos al lenguaje que sean el correcto funcionamiento del servicio.

Diagrama

Descripción generada automáticamente

Ilustración 4.2.1 - Modelo de clases de diseño

## 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 tres 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 | * Dirección IP * Direcciones IP * Rango de IPs * Red |
| user | Usuario de acceso a las máquinas a analizar | String | Nombres de usuarios |
| password | Contraseña del usuario de acceso | String | Contraseñas de usuarios |
| time | Tiempo de espera entre análisis en horas | Integer | [0 – 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 establecer 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 | * Fedora * Debian * SUSE |
| Servicios Instalados | Número de servicios que se encuentran instalados en la máquina. | Integer | [1 – N] |
| Servicios Actualizados | Número de servicios que han sido actualizados en la máquina durante el análisis. | Integer | [0 – N] |
| Servicios OK | Número de servicios que disponen de la última versión disponible instalada en la máquina. | Integer | [1 – N] |
| Servicios NOK | Número de servicios que no disponen de la última versión disponible instalada en la máquina. | Integer | [0 – N] |

Tabla 3 - Descripción de los datos resultantes de la ejecución

## 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 podrá 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 de *systemctl*.

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.

Interfaz de usuario gráfica

Descripción generada automáticamente

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 preciosa 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 servicios, dentro un periodo de tiempo concreto marcado por el usuario (Ilustración 4.4.3)

Interfaz de usuario gráfica, Gráfico, Aplicación

Descripción generada automáticamente

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

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

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

El dashboard se encuentra soportado sobre una interfaz web y es solo accesible desde la máquina donde se realiza el análisis, siempre y cuando el servicio esté habilitado.

## 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 una máquina con sistema operativo MacOS, se ha hecho uso de su terminal UNIX preinstalada. Esta se ha 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 hit en un repositorio en línea.
* **AWS**. Servicio web, proporcionado por Amazon, en el que se han ejecutado varias instancias de los diferentes sistemas operativos qué cubre el servicio. Al contrario que el resto de las herramientas, sí que se ha requerido de un pago para su uso.

## 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 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. La conexión a las máquinas o el registro de los parámetros de configuración 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 de debe de contar los test mínimos necesarios para cerciorase de que las pruebas sean válidas.

Todos los test unitarios se encuentran implementados en el fichero 'services-test.py’, pudiendo ser ejecutados, y tienen como función validar de una forma tangible la implementación correcta de los requisitos previamente planteados. Estas pruebas son:

|  |  |  |  |
| --- | --- | --- | --- |
| ID Req. | ID Test | Nombre | Resultado |
| RF1 | TS01 | Test lanzamiento servicio en máquina sin Python 3 | **OK.** Al lanzar el servicio se obtiene un error al no disponer del intérprete de Python 3 |
| RF1 | TS02 | Test lanzamiento servicio en máquina con Python 3 | **OK.** Al lanzar el servicio no se obtiene error alguno |
| RF1 | TS03 | Test comprobación máquina analizadora sin la herramienta systemctl | **OK.** Se comprueba que el servicio da error si no se dispone de la herramienta Systemd |
| RF1 | TS04 | Test comprobación máquina analizadora con la herramienta systemctl | **OK**. Se comprueba que el servicio no da error |
| RF1 | TS05 | Test comprobación máquina analizadora no UNIX | **OK.** Se comprueba que el servicio da error si la máquina no es UNIX |
| RF1 | TS06 | Test comprobación máquina analizadora UNIX | **OK.** Se comprueba que el servicio no da error |
| RF2 | TS07 | Test comprobación conectividad en máquina no levantada | **OK.** Se comprueba, mediante un paquete que el servicio no da error |
| RF2 | TS08 | Test comprobación conectividad en máquina levantada | **OK.** Se comprueba, mediante paquetes ICMP, que una máquina al probar la conexión da error |
| RF2 | TS09 | Test obtención sistema operativo máquina | **OK.** Se comprueba, mediante paquetes ICMP, que una máquina al probar la conexión no da error |
| RF2 | TS10 | Test comprobación sistema operativo máquina fuera del alcance | **OK.** Se comprueba que si la máquina no esta dentro del alcance, el servicio da error |
| RF2 | TS11 | Test comprobación sistema operativo máquina dentro del alance | **OK.** Se comprueba que si la máquina esta dentro del alcance, el servicio no da error |
| 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 a analizar |
| RF3 | TS13 | Test comprobación error puerto | **OK.** Se comprueba que se da Falso si no se introduce un puerto correcto |
| RF3 | TS14 | Test comprobación puerto SSH abierto en la máquina analizada | **OK.** Se comprueba que se llega al puerto 22 SSH (cerrado por defecto) de la máquina a analizar |
| RF4 | TS15 | Test obtención de un servicio preinstalado | **OK.** Al lanzar el escaneo y buscar un servicio preinstalado (crond), se obtiene el servicio |
| RF4 | TS16 | Test obtención de un servicio no instalado | **OK.** Al lanzar el escaneo y buscar un servicio no instalado (SSH), no se obtiene el servicio |
| RF5 | TS17 | Test acción servicio vacío | **OK.** Se comprueba que si el servicio analizado este vacío, no se realiza nada |
| RF5 | TS18 | Test acción servicio con versión actualizada | **OK.** Se comprueba que si el servicio analizado esta actualizado, no se realiza nada |
| RF5 | TS19 | 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 | TS20 | 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 | TS21 | Test obtención instrucciones para máquinas fuera del alcance | **OK.** Se comprueba que no se obtienen instrucciones |
| RF7 | TS22 | Test obtención instrucciones para máquinas dentro del alcance | **OK**. Se comprueba que se obtienen las instrucciones del sistema operativo |
| RF11 | TS25 | Test obtención máquina sin IP indicada | **OK**. Se comprueba que no se obtiene ninguna IP |
| RF11 | TS26 | Test obtención máquina de una IP | **OK**. Se comprueba que se obtiene una lista con la IP |
| RF11 | TS27 | Test obtención máquinas de una lista de IPs | **OK.** Se comprueba que se obtiene una lista con las IPs |
| RF11 | TS28 | Test obtención máquinas de un rango de IPs | **OK.** Se comprueba que se obtiene una lista con el rango de IPs |
| RF11 | TS29 | Test obtención máquinas de una IP | **OK.** Se comprueba que se obtiene una lista con las IPs de la red |
| RF13 | TS30 | Test instalación ficheros | **OK.** Se comprueba que los ficheros del servicio se encuentran en la máquina tras ejecutar el script |
| RF14 | TS31 | Test desinstalación ficheros | **OK.** Se comprueba que los ficheros del servicio no se encuentran en la máquina tras ejecutar el script |
| RF16 | TS32 | Test instanciación dashboard | **OK.** Se comprueba que el dashboard se inicia al habilitar el servicio |
| RF17 | TS33 | Test dashboard cuando el servicio está deshabilitado | **OK.** Se comprueba que el dashboard no se instancia si el servicio está deshabilitado |
| RF19 | TS34 | 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 | TS35 | Test comprobación fichero originan vacío tras reinicio | **OK**. Se comprueba que el fichero está vacío |
| RNF2 | TS36 | Test permisos ficheros de configuración | **OK.** Se comprueba que el usuario solo tienes permisos de lectura y escritura |
| RNF3 | TS37 | Test permisos scripts | **OK**. Se comprueba que el usuario solo tienes 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 para validar el resto de requisitos, un entorno de prueba el cual comprobará el funcionamiento del servicio. Para esta prueba, harán uso de 4 máquinas, 3 sistemas, para realizar el análisis de sus servicios; y en 1 sistema para ejecutar el servicio. Las máquinas por analizar contarán con un sistema operativo distinto, de los incluidos en el alcance del proyecto; y con 6 servicios preinstalados que 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.

Diagrama

Descripción generada automáticamente

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).

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 4.6.2 - Dashboard ejecución prueba escenario

Texto

Descripción generada automáticamente

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.

## Diagrama de infraestructuras de nivel 3

En el diagrama (Ilustración 4.7.1) se muestra una representación de la infraestructura de nivel tres, nivel 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.

Imagen que contiene reloj, pantalla, medidor

Descripción generada automáticamente

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 envió 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.

# Construcción

## Referencia al repositorio de software

El software desarrollado para resolución del proyecto se encuentra almacenado en el siguiente repositorio de GitHub <https://github.com/josemanuel179/tfg>.

## Manuales

### Módulos necesarios

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

* Paramito 2.10.4, para la ejecución de conexiones SSH entre máquinas.
* Dash 2.4.1, para la creación del dashboard interactivo.
* Plotly 5.8.0, para la creación de gráficas dentro del dashboard.
* Pandas 1.4.2, 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 los módulos en su versión necesaria.

### Instalación del servicio

El proceso de instalación del servicio es muy sencillo, solo se **debe ejecutar el fichero install.sh** desde una terminal de la siguiente forma:

*./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 Ilustración 5.2.2.1.

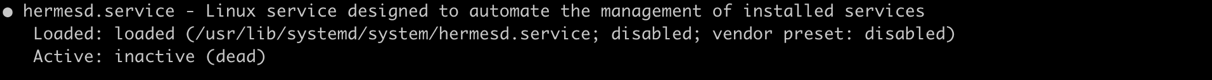


Ilustración 5.2.1 - Resultado instalación servicio

### 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 de acceso 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 siempre 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.

Texto

Descripción generada automáticamente

Ilustración 5.2.2 - Levantamiento del servicio

### 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:

*./remove.sh*

### 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 buscar la siguiente dirección:

[*http://127.0.0.1:8020*](http://127.0.0.1:8020)

Toda esta información, se encuentra almacenada en el mismo repositorio que el código. Específicamente, en el enlace [https://github.com/josemanuel179/ tfg#readme](https://github.com/josemanuel179/%20tfg#readme).

# 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 remarcar 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 en estos.

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.

# 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

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 Flanagal 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/

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/

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

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-338bfcb6ffa4

Tyler, C. (2006). *Fedora Linux: A Complete Guide to Red Hat’s Community Distribution.* O’Reilly Media, Inc.