Plugin GLPI – NVD

Escaneo de activos en busca de vulnerabilidades publicadas.

Universitat Oberta de Catalunya

Nombre Estudiante
Carlos Borau González

Nombre del Programa
Máster Universitario en Ciberseguridad y Privacidad

Área de trabajo final
Seguridad Empresarial

Nombre Tutor/a de TF
Borja Guaita Pérez

Profesor responsable de la asignatura
Víctor García Font

Fecha Entrega
13/06/2023

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Plugin GLPI – NVD*
Nombre del autor:	*Carlos Borau González*
Nombre del consultor/a:	*Borja Guaita Pérez*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	*06/2023*
Titulación o programa:	Máster Universitario en Ciberseguridad y Privacidad
Área del Trabajo Final:	*Seguridad Empresarial*
Idioma del trabajo:	*Castellano*
Palabras clave:	*Vulnerabilidades, GLPI, plugin*

Resumen del Trabajo

GLPI es una herramienta de escritorio de código abierto para la gestión de activos informáticos diseñada para permitir a desarrolladores externos ampliar la funcionalidad base de la aplicación a través de plugins. Esta herramienta permite llevar, de forma automatizada, un inventariado de equipos informáticos, sus características, y programas instalados en ellos, lo que resulta de gran utilidad a la hora de gestionar grandes redes informáticas y mantener actualizadas las componentes de estas redes, y las aplicaciones que emplean, evitando así potenciales incidentes relacionados con vulnerabilidades presentes en su código.

Este proyecto se centra en desarrollar un plugin para la herramienta GLPI que permita, a partir de la información recogida en el inventariado de la herramienta, identificar las distintas instalaciones de programas en los equipos gestionados por GLPI y buscar vulnerabilidades conocidas relacionadas con estas instalaciones en bases de datos que recogen registros de vulnerabilidades informáticas públicas. Si bien han existido intentos previos de aportar esta capacidad a la herramienta GLPI, actualmente se encuentran descontinuados y no existen planes para reanudar su desarrollo.

El desarrollo del proyecto ha estado marcado principalmente por las fases de investigación e implementación, divididas según las funcionalidades a implementar planteadas en la etapa de planificación. Pese a haber surgido algunas dificultades en el proceso, una vez finalizadas estas fases, se ha comprobado la capacidad del plugin desarrollado para recopilar y mostrar información acerca de vulnerabilidades presentes en los sistemas operativos y programas instalados en los equipos gestionados por GLPI.

Abstract

GLPI is an open-source desktop tool for IT asset management designed to allow external developers to extend its base functionality through plugins. This tool is capable of automatically keeping an inventory of computer equipment, its characteristics, and programs installed on it, which is proven to be significantly useful when managing large computer networks and maintaining the components of this networks and the applications they use updated, thus avoiding potential incidents related to vulnerabilities present in their code.

This project is focused on the development a plugin for GLPI that, provided the information collected in the tool's inventory, can identify the different program installations on the computers managed by GLPI and then is able to search for known vulnerabilities related to the identified installations in databases that store public IT vulnerability records. While some previous attempts to implement this functionality for the GLPI framework have been made, such projects are, to this date, discontinued and no plans to resume development on them have been announced.

The development of the project has mainly revolved around the research and implementation stages, which have been divided according to the functionalities to be implemented as outlined in the planning stage. Although some difficulties have arisen along the development process, once these stages were completed, the capability of the developed plugin to gather and display information pertaining to those vulnerabilities present in the operating systems and programs installed on the equipment managed by GLPI has been verified.

Índice

1. Introducción ... 1
1.1. Contexto y justificación del Trabajo ... 1
1.2. Objetivos del Trabajo ... 2
1.3. Impacto en sostenibilidad, ético-social y de diversidad ... 3
1.4. Enfoque y método seguido ... 4
1.5. Planificación del Trabajo ... 5
1.6. Seguimiento de la planificación ... 8
1.7. Breve sumario de productos obtenidos ... 9
1.8. Breve descripción de los otros capítulos de la memoria ... 9
2. Conceptos previos ... 10
2.1. GLPI ... 10
2.2. Vulnerabilidades públicas ... 11
2.3. Common Platform Enumeration (CPE) ... 12
3. Estado del arte ... 13
3.1. IVA (Inventory Vulnerability Analysis) ... 13
3.2. CVE Plugin for GLPI ... 13
4. Diseño e implementación del plugin ... 14
4.1. Preparación del entorno de desarrollo ... 14
4.1.1. Documentación e instalación de GLPI ... 14
4.1.2. Creación de un repositorio ... 14
4.1.3. Creación del esqueleto del plugin ... 14
4.2. Consultas a bases de datos de vulnerabilidades públicas ... 16
4.2.1. Elección de base de datos pública ... 16
4.2.2. Clases creadas para las consultas ... 16
4.3. Tareas automáticas ... 18
4.3.1. Definición de la tarea ... 18
4.3.2. Registro de la tarea ... 18
4.3.3. Funcionalidades adicionales de la tarea automática ... 19
4.4. Persistencia en base de datos local ... 20
4.4.1. Tabla glpi_plugin_nvd_vulnerabilities ... 20
4.4.2. Tabla glpi_plugin_nvd_vulnerability_descriptions ... 21
4.4.3. Tabla glpi_plugin_nvd_vulnerability_configurations ... 22
4.4.4. Tabla glpi_plugin_nvd_vulnerable_versions ... 23
4.4.5. Tabla glpi_plugin_nvd_vulnerable_system_versions ... 23
4.4.6. Tabla glpi_plugin_nvd_cpe_software_associations ... 24
4.4.7. Tabla glpi_plugin_nvd_config ... 26
4.4.8. Consistencia de la base de datos ... 26
4.4.9. Clases que acceden a la base de datos ... 26
4.4.10. Esquema de las principales tablas de la base de datos ... 27

4.5. Inventariado de activos informáticos 28
4.5.1. Comparación entre resultados del inventariado y estándar CPE . 29
4.6. Creación de vistas y generación de alertas..... 30
4.6.1. Configuración del plugin 30
4.6.2. Asociación entre programa y nombres de fabricante y producto CPE 31
4.6.3. Visualización de vulnerabilidades..... 33
4.6.3.1. Visualización conjunta de vulnerabilidades 34
4.6.3.2. Visualización de vulnerabilidades por equipo..... 36
4.6.3.3. Visualización de vulnerabilidades por programa 38
4.6.4. Clases de soporte..... 39
4.7. Dificultades encontradas..... 40
4.7.1. Identificación de programas y sistemas operativos 40
4.7.2. Registros de vulnerabilidades sin versión especificada..... 42
5. Pruebas y resultados..... 43
5.1. Consultas a base de datos NVD 43
5.2. Interacción con la base de datos local 43
5.3. Procesado del inventariado de GLPI..... 44
5.4. Vistas 46
5.5. Pruebas finales 46
6. Conclusiones y trabajos futuros 48
6.1. Conclusiones principales..... 48
6.2. Trabajo a futuro..... 49
7. Glosario..... 51
8. Bibliografía 52
9. Anexos 54
9.1. Instalación y configuración del plugin..... 54

Lista de figuras

Figura 1: Objetivos de Desarrollo Sostenible (ODS) 8, 9 y 10..... 3
Figura 2: Diagrama de Gantt del proyecto. 7
Figura 3: Estructura del plugin nvd en el sistema de ficheros. 15
Figura 4: Estructura de la tabla glpi_plugin_nvd_vulnerabilities. 21
Figura 5: Estructura de la tabla glpi_plugin_nvd_vulnerability_descriptions..... 21
Figura 6: Estructura de la tabla glpi_plugin_nvd_vulnerability_configurations. 22
Figura 7: Estructura de la tabla glpi_plugin_nvd_vulnerable_software_versions. 23
Figura 8: Estructura de la tabla glpi_plugin_nvd_vulnerable_system_versions. 24
Figura 9: Estructura de la tabla glpi_plugin_nvd_cpe_software_associations.. 25
Figura 10: Estructura de la tabla glpi_plugin_nvd_config. 26
Figura 11: Esquema de las principales tablas de la base de datos de GLP. Generado con la herramienta dbdiagram.io [23]. 27
Figura 12: Importación del archivo JSON de inventariado en GLPI. 28
Figura 13: Vista de configuración del plugin NVD. 31
Figura 14: Vista de asignación de nombres de fabricante y producto CPE para un programa. 32
Figura 15: Vista tras la asignación de nombres CPE realizada. 33
Figura 16: Vista general de vulnerabilidades de programas. 35
Figura 17: Vista general de vulnerabilidades de sistemas operativos. 35
Figura 18: Alerta mostrada al hacer click en un aviso. 36
Figura 19: Vista de vulnerabilidades de programas instalados en un dispositivo. 37
Figura 20: Vista de vulnerabilidades del sistema operativo de un dispositivo. . 37
Figura 21: Alerta de vulnerabilidad con restricciones para un sistema operativo concreto..... 37
Figura 22: : Vista de vulnerabilidades de las versiones de un programa. 38
Figura 23: Alerta de vulnerabilidad con restricciones para un programa concreto. 38
Figura 24: Estadísticas de identificación de sistemas operativos empleando los nombres presentes en los informes de inventariado. 45
Figura 25: Estadísticas de identificación de aplicaciones empleando los nombres presentes en los informes de inventariado. 45
Figura 26: Estadísticas de identificación de sistemas operativos empleando el mecanismo de traducción implementado. 46
Figura 27: Estadísticas de identificación de aplicaciones empleando los nombres CPE de las asociaciones creadas. 46
Figura 28: Distribución de las vulnerabilidades encontradas según el tipo de configuración a la que afectan..... 47
Figura 29: Información de las vulnerabilidades encontrades para la muestra.. 47

1. Introducción

1.1. Contexto y justificación del Trabajo

A lo largo de los últimos años la ciberseguridad se ha consolidado como una de las disciplinas de mayor importancia dentro del ámbito de las tecnologías de la información, ya sea en su aplicación dentro de las infraestructuras informáticas de empresas como en su uso por particulares. Esta creciente relevancia es en buena parte debida al traslado de diversas actividades, tanto profesionales como cotidianas, al plano informático, y de la subsecuente necesidad de gestionar de forma segura los equipos informáticos empleados en estas actividades y la información que estos tratan y almacenan. Con este fin han surgido múltiples tecnologías que pretenden facilitar a los usuarios la gestión de los activos informáticos de los que disponen o los cuales están encargados de administrar, ofreciéndoles en una sola aplicación capacidades integradas para el inventariado, aprovisionado y control de los programas, conexiones o archivos presentes los equipos gestionados.

Dentro de este contexto nace GLPI [1], una herramienta de código abierto para la gestión de activos informáticos en empresas que, además de sus capacidades de administración e inventariado, ofrece la posibilidad de integrar extensiones del software base (plug-ins) programadas por terceros para añadir nuevas funcionalidades a la aplicación. Pese a que GLPI no es una herramienta concebida para aportar seguridad a la infraestructura informática de una empresa, existen plug-ins diseñados para permitir la interacción entre su framework y diversas aplicaciones de ciberseguridad [2], así como proyectos que en su momento buscaron implementar en GLPI la detección de vulnerabilidades conocidas a través de la consulta a bases de datos públicas [3] [4]. El desarrollo de estos últimos proyectos, sin embargo, ha sido o bien abandonado o puesto en parada indefinida, situación que se ha mantenido hasta la actualidad, por lo que la interesante funcionalidad que propusieron añadir al framework de GLPI permanece sin haberse implementado.

Así pues, este proyecto se centrará en el desarrollo de un plug-in para la herramienta GLPI que permita a los administradores de una infraestructura informática detectar de forma ágil vulnerabilidades en sus sistemas basándose en su identificación a través de las versiones de los programas instalados en estos y su comparación con los registros de vulnerabilidades conocidas de tales programas. De igual manera se busca poder integrar la capacidad para generar vistas para la gestión de las alertas surgidas por las vulnerabilidades detectadas, así como la categorización de estas en función de su criticidad.

1.2. Objetivos del Trabajo

El objetivo principal de este proyecto es el de desarrollar un plugin para la herramienta GLPI cuya principal funcionalidad sea la de identificar vulnerabilidades software en los equipos de la infraestructura informática gestionada por la aplicación, así como la categorización de estas vulnerabilidades en función de su criticidad y la presentación de estas de forma que facilite su gestión y resolución de forma ágil. Este objetivo puede dividirse en la siguiente lista de subobjetivos:

- Familiarización con el framework de GLPI, instalación, configuración y administración de la herramienta.

- Familiarización con las herramientas de desarrollador de GLPI, API ofrecida, guías de desarrollo de plug-ins y estándares y lenguajes de programación.

- Investigación acerca de las diferentes bases de datos públicas de vulnerabilidades conocidas, métodos de consulta, formato de datos y periodos de actualización.

- Instalación de un agente de inventariado para facilitar la identificación del software instalado en los equipos gestionados por GLPI.

- Implementación de la interacción entre GLPI y la base de datos de vulnerabilidades conocidas escogida. Almacenamiento local de las vulnerabilidades consultadas.

- Implementación de la detección de vulnerabilidades en los equipos gestionados por GLPI en función de la información almacenada a cerca de vulnerabilidades conocidas.

- Implementación de la clasificación de las vulnerabilidades en función de su criticidad por la clase de vulnerabilidad y el tipo de equipo en el que hayan sido encontradas.

- Implementación de vistas para la presentación de las vulnerabilidades identificadas que permitan facilitar la gestión y resolución de las mismas de forma ágil.

- Exponer y publicar los resultados y conclusiones del trabajo para que sean accesibles de forma libre, así como definir posibles vías de trabajo a futuro que se puedan desprender de este proyecto.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Debido a que el objetivo final del proyecto es el desarrollo y publicación de una extensión de la funcionalidad de una herramienta de software libre, podría parecer que el impacto de este en dimensiones como la sostenibilidad o la diversidad pudiera ser mínimo. No obstante, desde la concepción de este proyecto se ha tenido en cuenta su potencial para mejorar la eficiencia de los procesos internos de las empresas que requieran herramientas para gestionar de forma segura sus activos, así como la disponibilidad del mismo para cualquier usuario final sin importar su capacidad económica, al ser de código abierto.

En el plano ético-social es quizás donde más impacto podría llegar a tener el proyecto desarrollado, debido principalmente al ámbito en el que se enmarca, esto es, la ciberseguridad. En general, las herramientas desarrolladas para permitir a los usuarios mejorar su estrategia en defensa y detección contra amenazas informáticas, contribuyen de manera activa en proteger a estos usuarios, a sus equipos informáticos y a la información almacenada en estos, de conductas, muchas veces criminales, contra estos activos.

Teniendo estas aproximaciones en cuenta puede entenderse como el proyecto desarrollado se enmarca en los ODS 2030 de la ONU [5], principalmente alineándose con la meta 9.1 pero también con la 8.2 o 10.2.

Figura 1: Objetivos de Desarrollo Sostenible (ODS) 8, 9 y 10.

1.4. Enfoque y método seguido

El enfoque y la metodología a seguir en este proyecto pueden dividirse en cuatro etapas claramente diferenciadas:

- Etapa de planificación: Definición de objetivos del proyecto, metodología a emplear para cumplirlos, así como división y programación de las diferentes tareas a realizar a lo largo del proyecto. Esta etapa debería concluirse con la entrega de la PEC1, es decir, el plan de trabajo.

- Etapa de investigación: Estudio previo del estado del arte, tecnologías a emplear, la consulta de su documentación pública en lo referente a interacción (API), integración, lenguajes empleados y requisitos de operabilidad, así como de un análisis de su idoneidad y aplicabilidad para lograr desarrollar los objetivos del proyecto.

- Etapa de implementación: Una vez escogidas las soluciones más adecuadas para llevar a cabo el proyecto, se procederá a la implementación de la extensión de funcionalidad de GLPI, que se ha definido como el objetivo final del mismo. En cada aspecto o funcionalidad de la extensión implementada se procederá a aplicar las herramientas/tecnologías estudiadas que mejor permitan su programación.

- Etapa de resultados y redacción: Tras haber finalizado la implementación del plug-in de GLPI se procederá al análisis de los resultados obtenidos, así como a la redacción de la memoria final del proyecto, las conclusiones y los posibles trabajos a futuro que puedan derivarse del mismo. Esta etapa debería concluirse con la entrega de la PEC4.

Siguiendo este enfoque de división en etapas, a lo largo de las cuales se realizarán múltiples reuniones de seguimiento, se planea lograr cumplir con los objetivos principales definidos para el proyecto.

1.5. Planificación del Trabajo

A continuación, se detallan las principales tareas que componen cada una de las fases expuestas en el apartado anterior:

- **Documentación previa**: Estudio del contexto y el estado del arte en lo referente al tema del trabajo.

- **Definición del ámbito del TFM**: Se concreta la temática y alcance del trabajo.

- **Plan de trabajo**: Se define el contexto y justificación del trabajo, los objetivos a cumplir con el mismo, los objetivos de desarrollo sostenible con los que está relacionado, la metodología empleada en su desarrollo y las tareas a realizar, así como la planificación de estas.

- **Investigación sobre GLPI**: Estudio de la documentación disponible sobre la herramienta GLPI, instalación, administración, y capacidades para extender su funcionalidad, dentro de las cuales se incluyen:

    - Creación de tareas automatizadas.
    - Interacción con bases de datos.
    - Inventariado de activos informáticos.
    - Creación de vistas e interfaz gráfica.

- **Investigación sobre bases de datos de vulnerabilidades conocidas**: Estudio de las diferentes alternativas para consulta de vulnerabilidades conocidas, así como la descarga de esta información.

- **Implementación del esqueleto del plug-in**: Creación de la estructura en sistema de ficheros del plugin, así como de los principales archivos y métodos de instalación y desinstalación.

- **Implementación de consultas a la B.D**: Consiste en añadir al plugin la funcionalidad de realizar consultas a la base de datos escogida. Estas consultas pueden ser tanto manuales como automáticas.

- **Implementación de la persistencia en B.D**: Definición de un esquema relacional, operaciones y modelo de objetos que permitan al plug-in almacenar la información obtenida en una base de datos local.

- **Implementación del inventariado**: Instalación de un plug-in que permita realizar un inventariado de los activos gestionados por GLPI, más concretamente del software instalado en los mismos.

- **Generación de alertas**: Consiste en añadir al plug-in la capacidad para comparar las versiones del software instalado en las máquinas controladas desde GLPI, a través del plug-in de inventariado, con aquellas versiones vulnerables almacenadas en la base de datos local.

- **Retoques**: Posibles modificaciones necesarias previas a la entrega de seguimiento.

- **Conclusiones del proyecto**: Presentación de las conclusiones obtenidas de la realización del proyecto.

- **Trabajo a futuro**: Presentación de las posibles vías de trabajo a futuro desprendidas del proyecto realizado.

- **Redacción de la memoria final**: Composición del documento formal que recoge tanto los apartados anteriores como todos los aspectos relevantes de las diferentes fases del proyecto. Este documento cuenta con Índices, bibliografía y anexos.

- **Elaboración del vídeo presentación**: Grabación de un vídeo expositor de la funcionalidad de la extensión desarrollada.

Estas tareas, junto a las subtareas de las que se componen algunas de ellas, así como con las fechas de inicio, fin y duraciones absolutas en días de estas se encuentran reflejadas en la **Figura 2**.

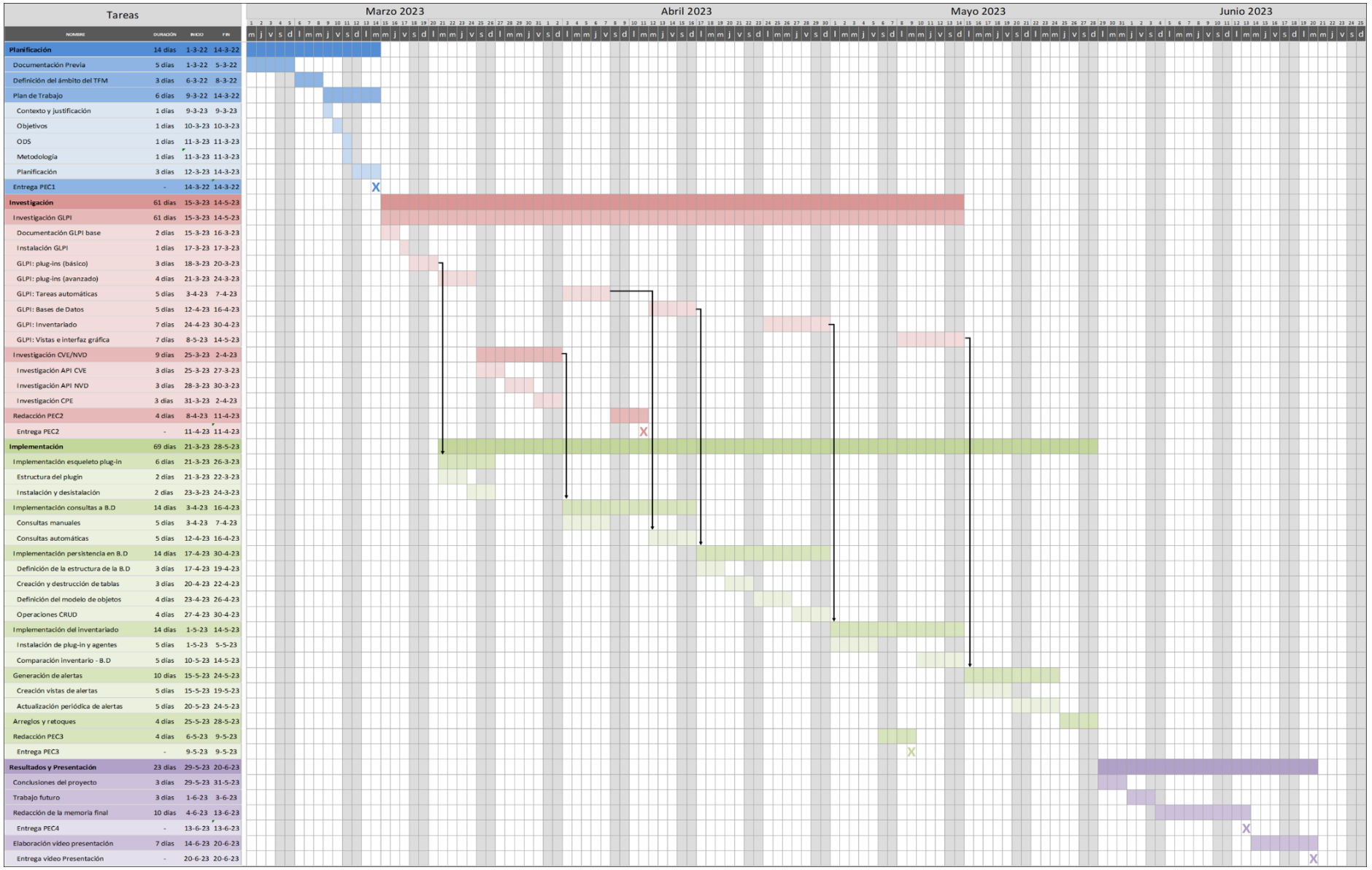

Figura 2: Diagrama de Gantt del proyecto.

1.6. Seguimiento de la planificación

En este apartado se detalla, para cada entrega realizada durante el desarrollo del proyecto, el estado de las tareas planificadas en el momento de la entrega junto a posibles cambios en la planificación y sus correspondientes justificaciones pertinentes:

- **PEC1 - Plan de trabajo**: A fecha 14/3/2023 se ha realizado la entrega de la PEC1, correspondiente al plan de trabajo del proyecto. Las tareas planificadas (Documentación, definición del ámbito del TFM y elaboración de la planificación) así como sus correspondientes subtareas, encuadradas todas en la etapa de planificación, se han podido llevar a cabo sin contratiempo alguno.

- **PEC2 - Entrega de seguimiento**: A fecha 11/4/2023 se ha realizado la entrega de la PEC2, correspondiente a la primera entrega de seguimiento del proyecto. Debido a dificultades a la hora de investigar el funcionamiento interno del framework de GLPI, así como la necesidad de abordar una nueva tarea de investigación que no se tuvo en cuenta al momento de realizar la planificación, no ha podido cumplirse con la totalidad de las tareas que se habían proyectado para esta entrega, faltando por implementar la automatización de las consultas a bases de datos.

  Se ha realizado un ligero ajuste sobre la planificación original para adaptar el desarrollo del proyecto a las nuevas circunstancias, incluyendo nuevas tareas de investigación y ampliando el plazo de las etapas de investigación e implementación, así como de sus tareas en líneas generales.

- **PEC3 - Entrega de seguimiento**: A fecha 9/5/2023 se ha realizado la entrega de la PEC3, correspondiente a la segunda entrega de seguimiento del proyecto. Para esta entrega se ha cumplido con la planificación finalizando con la investigación e implementación de las consultas automáticas, persistencia en base de datos local e inventariado de activos mediante el agente de GLPI.

- **PEC4 - Memoria final**: A fecha 13/6/2023 se ha realizado la entrega de la PEC4 (memoria final del proyecto). Para esta última entrega se ha cumplido con la planificación, finalizando con la comparación entre los reportes de inventariado obtenidos a través del agente de inventariado y las configuraciones disponibles en las bases de datos de vulnerabilidades, creando vistas para facilitar la interacción del usuario con el plugin y la visualización de vulnerabilidades y automatizando la eliminación de registros de vulnerabilidades anteriormente presentes en algún equipo gestionado por GLPI. Adicionalmente se ha añadido la funcionalidad de detectar vulnerabilidades basadas en una lista reducida de sistemas operativos. Por último, se han redactado las conclusiones y el trabajo a futuro y se ha finalizado la memoria.

1.7. Breve sumario de productos obtenidos

Como resultado del proyecto, se ha desarrollado un plugin para la herramienta de gestión de activos informáticos GLPI que permite, existiendo un inventariado previo de equipos con sus respectivas instalaciones de programas y sistemas operativos, recorrer dicho inventariado en busca de versiones de aplicaciones y sistemas vulnerables, obtener información de bases de datos de vulnerabilidades públicas acerca de las vulnerabilidades encontradas, y registrar esta información y las relaciones entre las vulnerabilidades y las versiones vulnerables para permitir su posterior visualización.

El desarrollo del plugin se ha centrado en obtener un producto que facilite al usuario la interacción y navegabilidad de forma intuitiva a la vez que proporcione una funcionalidad lo más fiel a los objetivos planteados del proyecto.

1.8. Breve descripción de los otros capítulos de la memoria

En el capítulo **2** se introducen una serie de conceptos previos relevantes para comprender el desarrollo del proyecto, su origen, sus objetivos, las dificultades que se han encontrado durante su realización y las decisiones que se han tomado para llegar al producto final. El capítulo **3** presenta el contexto del estado del arte en el que se enmarca el proyecto, mencionando los anteriores intentos, actualmente descontinuados, de implementar la funcionalidad que ofrece el plugin desarrollado. El capítulo **4**, siendo el capítulo más extenso, contiene las diferentes fases del diseño y la implementación del plugin. Cada una de las fases, como se definieron en la fase de planificación, está a su vez compuesta de múltiples subfases, las cuales presentan funcionalidades o aspectos implementados del plugin, con las correspondientes justificaciones y decisiones tomadas para su implementación. El final de este capítulo lista una serie de dificultades encontradas a lo largo de la implementación del plugin y los enfoques seguidos para abordarlas o evitarlas. En el capítulo **5** se detallan una serie pruebas realizadas durante el desarrollo del plugin, para comprender el funcionamiento de la herramienta GLPI y cómo implementar las funcionalidades definidas en la fase de planificación, y una vez finalizado el desarrollo, para comprobar la correcta implementación de las mencionadas funcionalidades. Las conclusiones del proyecto y trabajos futuros relacionados con este se detallan en el capítulo **0**. Por último, los capítulos **7**, **8** y **9** contienen el glosario, la bibliografía y los anexos de la memoria del proyecto.

2. Conceptos previos

2.1. GLPI

GLPI es una herramienta de escritorio de código abierto para la gestión de activos informáticos, tanto equipos como componentes de la red y software instalado en estos, que proporciona además la capacidad de crear "*issues*" sobre los que realizar un seguimiento. Para su funcionamiento es necesario que se instale en un equipo que cuente tanto con un servidor web PHP como con una base de datos de tipo MySQL.

Una vez instalada la herramienta, se puede acceder a la misma a través de un portal web mediante una cuenta de usuario, que previamente habrá de ser creada por los administradores, y a la cual se le deberán de asignar los privilegios correspondientes a su rol dentro de la entidad. Cada usuario cuenta con la posibilidad de personalizar las vistas y "*dashboards*" que ofrece la herramienta, así como de participar en el proceso de administración de los activos gestionados por GLPI en función de los privilegios que se le hayan otorgado.

Actualmente existen múltiples soluciones dirigidas a la administración de activos de una infraestructura informática en el mercado, sin embargo, algunas de las características que hacen a GLPI destacar sobre tales soluciones son:

- **Disponibilidad**: Al tratarse de una solución de código abierto el software se encuentra disponible de forma gratuita tanto para empresas como para particulares.
- **Interoperabilidad**: Debido a que los requerimientos para su funcionamiento son únicamente su instalación conjunta con un servidor web que soporte PHP y una base de datos de tipo MySQL, es posible instalar GLPI en máquinas con sistemas operativos diferentes (Windows, Linux, MacOS).
- **Soporte técnico**: Pese a no ser gratuito cabe destacar que GLPI ofrece soporte técnico con garantía ajustado a las necesidades de cada cliente.
- **Capacidad para extender la funcionalidad básica**: El framework de GLPI cuenta con la capacidad para soportar "*plugins*", es decir, componentes que añaden características o funcionalidades específicas a la herramienta original. Estas componentes pueden haber sido desarrolladas por el propio equipo de GLPI o por la comunidad de desarrolladores de la herramienta, al tratarse de una solución de código abierto.

  Existe un repositorio público de plugins [6] gestionado por el equipo de GLPI en el que pueden encontrarse, categorizados en diferentes secciones, diversos plugins, así como documentación de referencia para el desarrollo de nuevos plugins.

2.2. Vulnerabilidades públicas

Debido a la gran cantidad y variedad de software, hardware y protocolos desarrollados, publicados y actualizados cada año no es de extrañar que, prácticamente a diario, se encuentren vulnerabilidades en el diseño y/o implementación de aplicaciones y componentes informáticos que afectan a los activos de empresas y particulares de todo el mundo. Los principales objetivos a la hora de hacer pública una vulnerabilidad son, por un lado, notificar a los desarrolladores del software/hardware/protocolo afectados por la vulnerabilidad para facilitar su resolución en el menor tiempo posible y, por otro lado, alertar a la comunidad de consumidores informáticos de posibles amenazas vinculadas al uso de los productos afectados para que tomen medidas destinadas a protegerse mientas los desarrolladores se encargan de actualizarlos.

En este contexto, y para facilitar la publicación y seguimiento de vulnerabilidades publicadas, surgieron diferentes bases de datos públicas que recogen vulnerabilidades informáticas publicadas, tanto por desarrolladores como por miembros de la comunidad informática, junto a información detallada a cerca de su funcionamiento, productos afectados, versiones vulnerables, métricas de criticidad y sugerencias para su resolución. No todas estas bases de datos ofrecen el mismo tipo o cantidad de información sobre las vulnerabilidades registradas, ni la misma capacidad de consulta o filtrado sobre la información incluida en sus registros. Las dos bases de datos de vulnerabilidades públicas más relevantes son:

- **CVE MITRE**: El proyecto CVE [7] de MITRE fue lanzado en septiembre de 1999 con el objetivo de definir, identificar y catalogar vulnerabilidades de ciberseguridad publicadas. Sigue un estándar de identificación que ha sido empleado por bases de datos posteriores. Entre la información ofrecida por esta base de datos en cada registro de cada una de las vulnerabilidades que recoge se puede encontrar:
    - Identificador único CVE.
    - Breve descripción de la vulnerabilidad.
    - Referencias a páginas externas para ampliar la información sobre la vulnerabilidad.
    - Entidad certificada que ha solicitado el registro de la vulnerabilidad.
    - Fecha de creación del registro en la base de datos.

  La base de datos de CVE cuenta con una API REST [8] que permite obtener registros filtrando por tipo de producto, versión y plataforma a través de CPE [9] (Common Platform Enumeration). Además, la API también ofrece diferentes funcionalidades como:
    - Consulta de registros por identificadores CVE.
    - Consulta de listas de fabricantes y sus productos aceptados en el formato CPE.
    - Conversión de códigos CPE al formato estándar CPE 2.2 o 2.3.
    - Filtrado de los registros por restricciones temporales.

- **NVD NIST**: El proyecto NVD [10] (*National Vulnerability Database*), mantenido por el NIST (*National Institute of Standards and Technology*), es un repositorio para la gestión de vulnerabilidades informáticas e información relacionada con estas que se encuentra gestionado por el gobierno de los Estados Unidos y está sincronizado con CVE de forma que los identificadores empleados para cada vulnerabilidad son los mismos asignados por el proyecto CVE de MITRE. Los registros que almacena la base de datos de NVD contienen toda la información disponible a cerca de la vulnerabilidad que puede encontrarse en la base de datos de CVE y, de forma adicional, ofrece los siguientes datos:
    - Métricas de severidad: Estas métricas, que siguen el formato estándar CVSS [11] (*Common Vulnerability Scoring System*), se presentan en forma de vector cuyas componentes representan, de forma numérica y en rangos entre 0 y 10, la severidad de los diferentes aspectos de la vulnerabilidad a la que hacen referencia. En función de la valoración final de la vulnerabilidad se le asigna una severidad: *None* (0), *Low* (0.1 - 3.9), *Medium* (4.0 - 6.9), *High* (7.0 – 8.9) o *Critical* (9.0 – 10).
    - Referencias a páginas externas para obtener recomendaciones sobre soluciones o mitigaciones para la vulnerabilidad o herramientas que puedan ayudar en su subsanación.
    - Identificador CWE [12] (*Common Weakness Enumeration*): Este identificador permite clasificar las vulnerabilidades en función de su modo de manifestarse o explotarse. El estándar se encuentra mantenido por MITRE.
    - Configuraciones afectadas: Lista de aplicaciones/sistemas operativos/hardware afectados por la vulnerabilidad. Cada componente de esta lista ha de seguir el formato CPE, actualmente de su versión 2.3, para identificar de forma inequívoca los componentes afectados.

  La base de datos NVD cuenta con una API [13] que permite filtrados más complejos a los de CVE. Estos filtrados pueden realizarse mediante identificadores CVE, CWE, métricas CVSS, códigos CPE, versiones, parámetros temporales de publicación y modificación de un registro o incluso cadenas de texto plano o expresiones regulares.

2.3. Common Platform Enumeration (CPE)

CPE es un sistema de nomenclatura estructural para sistemas informáticos y software que permite la identificación inequívoca de estos productos, así como su filtrado según una serie de reglas. Este formato fue desarrollado por el NIST y actualmente se mantiene documentación pública en la que puede consultarse la última versión del estándar (2.3), el formato de nomenclatura y un diccionario con ejemplos de componentes hardware y aplicaciones y sistemas operativos.

3. Estado del arte

Pese a la abundante oferta de plugins de diversas funcionalidades para la herramienta GLPI, incluyendo múltiples plugins que integran en el framework de GLPI la interacción con aplicaciones o escáneres de seguridad, tan sólo han existido dos que, en su momento, trataran de implementar un análisis de vulnerabilidades conocidas sobre los activos informáticos gestionados por GLPI. Ninguno de estos plugins, sin embargo, ha continuado su ciclo de vida hasta la actualidad, ya sea por falta de mantenimiento o por un desarrollo incompleto, por lo que el objetivo final de este proyecto pasa por suplir este interesante vacío de funcionalidad para la herramienta GLPI.

3.1. IVA (Inventory Vulnerability Analysis)

Se trata de un sistema modular desarrollado en Python [3] con la capacidad de comparar el inventariado de software gestionado por GLPI con los productos y versiones con vulnerabilidades publicadas en la base de datos CVE a través de CPE. Para ello, cada uno de sus módulos implementa diferentes funcionalidades (Obtención del inventario software de GLPI, Generación de CPEs, Interacción con CVE, Generación de alertas, acceso seguro…) y estos se interconectan para completar la solución.

Estrictamente hablando no se trata de un plugin para GLPI, pues únicamente emplea la funcionalidad de inventariado de la herramienta sin instalarse en esta. Esta solución se encuentra descontinuada desde 2017 y actualmente no es compatible con las últimas versiones de GLPI.

3.2. CVE Plugin for GLPI

Este plugin [4], desarrollado para las versiones entre 9.5.0 y 9.6.0 de GLPI, implementa, de forma interna a la herramienta, la obtención de productos y versiones gestionados por GLPI y su comparación contra la base de datos de CVE, mediante CPE, en busca de vulnerabilidades publicadas. Su desarrollo se encuentra actualmente en parada indefinida desde 2021, por lo que no ha sido publicado en el repositorio oficial de plugins de GLPI al no haber finalizado la implementación de las principales funcionalidades del plugin.

4. Diseño e implementación del plugin

4.1. Preparación del entorno de desarrollo

Antes de comenzar con la implementación del plugin se ha procedido a preparar el entorno empleado para el desarrollo de este. En este apartado se detallan las distintas fases de la preparación.

4.1.1. Documentación e instalación de GLPI

El primer paso para aproximarse al desarrollo de un plugin para una herramienta software ha de ser consultar la documentación general [14] y de desarrollador [15] [16] de la herramienta. Una vez se posee una idea clara sobre el funcionamiento de la misma y sus capacidades y limitaciones puede empezar el proceso de diseño e implementación. Para poder trabajar con la herramienta es necesario instalarla en un equipo siguiendo la documentación antes mencionada que, en este caso, indica la necesidad de disponer de un servidor web compatible con PHP, así como una base de datos tipo MySQL en la máquina en la que se realiza la instalación. Como ejemplo, para la instalación, se ha empleado XAMPP [17], un entorno de desarrollo libre que integra, entre otras tecnologías, PHP y una base de datos MariaDB.

4.1.2. Creación de un repositorio

Para gestionar de forma eficiente el desarrollo y versionado del código del plugin a desarrollar se ha creado un repositorio público en GitHub [18] al cual se subirán las actualizaciones realizadas sobre el plugin y desde el cual estará disponible para su descarga y posterior instalación en cualquier equipo que emplee GLPI.

4.1.3. Creación del esqueleto del plugin

Siguiendo la documentación de desarrolladores de GLPI se ha creado bajo el directorio de plugins de GLPI una carpeta sincronizada con el repositorio de GitHub antes mencionado y, en esta, la estructura de carpetas y ficheros necesarios para la detección del plugin por parte de GLPI para su correcta instalación y definición y estructuración de clases, con sus dependencias, y scripts que aportarán la funcionalidad a desarrollar al plugin.

Los únicos ficheros estrictamente necesarios a crear son, en primer lugar, **setup.php**, que contendrá la definición de información acerca del plugin (Nombre, Compatibilidad con versiones de GLPI, Autor, Módulos de PHP requeridos…), los registros de las clases creadas junto a las vistas que, de ser necesario, modificarán en la aplicación base y los “hooks” definidos, y, en segundo lugar, **hook.php**, en el que se implementarán los “hooks” de instalación y desinstalación (Creación/borrado de tablas en la base de datos, registro de tareas automáticas…). Adicionalmente, algunos ficheros relevantes a añadir son:

- **LICENSE**: Licencia GNU GPLv3 para el plugin desarrollado.
- **README.md**: Fichero que describe las características del plugin y su método de instalación.
- **nvd.png** y **nvd.xml**: Imagen y fichero de identificación del plugin para el repositorio de plugins de GLPI.

Por otro lado, los principales directorios requeridos en la estructura del plugin son, **inc**, donde se almacenarán los archivos php que contengan las clases definidas para el funcionamiento del plugin, y **front**, que contendrá los scripts php destinados a responder a peticiones internas de formularios. Además de estos directorios se han creado las siguientes carpetas:

- **css**: Contiene, en forma de ficheros css, definiciones de estilo para las vistas html creadas en el plugin.
- **js**: Contiene, en forma de ficheros js, funciones de javascript empleadas para dar dinamismo a las vistas html creadas en el plugin.
- **installation**: Contiene los scripts php que se encargan del proceso de instalación, desinstalación o actualización del plugin. Implementan los métodos empleados desde los "hooks" de instalación/desinstalación.

Así pues, la estructura final del plugin en el sistema de ficheros, con las carpetas y ficheros mencionados, puede observarse en la **Figura 3**.

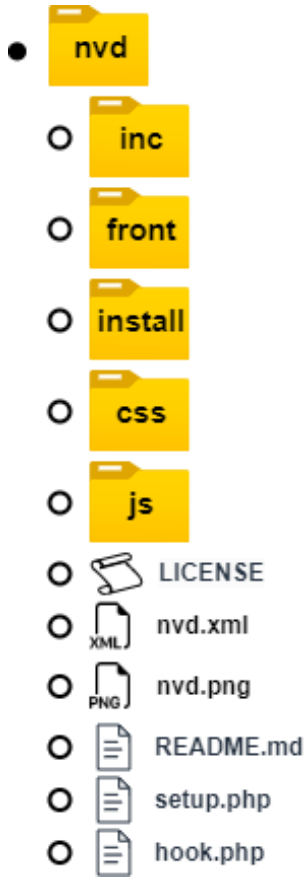

Figura 3: Estructura del plugin nvd en el sistema de ficheros.

4.2. Consultas a bases de datos de vulnerabilidades públicas

La primera funcionalidad que se ha decidido implementar ha sido la de la consulta de las bases de datos de vulnerabilidades públicas, en primera instancia de forma manual. Para ello se han estudiado las APIs ofrecidas por las dos principales bases de datos de vulnerabilidades públicas (CVE y NVD) y se han desarrollado una serie de clases PHP para soportar esta funcionalidad en el plugin diseñado.

4.2.1. Elección de base de datos pública

Para escoger la base de datos de la que obtener registros de vulnerabilidades se han tenido en cuenta, principalmente, la capacidad de consulta y filtrado que ofrecen sus APIs y la información y detalle que contienen los registros obtenidos como respuesta a las consultas.

- **CVE**: En el caso de la API de consultas de la base de datos de CVE [8] podemos destacar el requerimiento de una autenticación para poder realizar las consultas, mediante un usuario y una contraseña, token o id de sesión. Por otro lado, si bien nos permite filtrar por producto, la cantidad de información aportada para cada vulnerabilidad es muy limitada.
- **NVD**: En el caso de la API de consultas de la base de datos NVD [13] podemos destacar que únicamente requiere una "*apiKey*" para llevar a cabo las consultas (esta puede solicitarse al NIST), permite, al igual que el caso anterior, filtrar las vulnerabilidades por el producto y versión, pero además ofrece más información acerca de las vulnerabilidades que devuelve (CVSS, CWE, CPE…).

Debido a la mayor facilidad de consulta y cantidad de información sobre las vulnerabilidades que ofrece NVD y, ya que esta se encuentra enlazada con la base de datos CVE, conteniendo todas las vulnerabilidades confirmadas en esta, se ha escogido emplear NVD como la base de datos de la que obtener registros de vulnerabilidades para el plugin.

De cara a implementar ciertas funcionalidades del plugin, como la ordenación de vulnerabilidades en función de su severidad o la identificación de los componentes afectados y las características determinantes de estos (edición, sistema operativo o arquitectura objetivo), la información adicional aportada por NVD resulta clave.

4.2.2. Clases creadas para las consultas

Bajo el directorio **inc** en la estructura del plugin en el sistema de ficheros se han definido una serie de clases, cada una de las cuales implementa un aspecto de la interacción entre la aplicación GLPI y la API de la base de datos NVD. Estas clases son dependientes entre ellas y algunas de ellas han sido modificadas en posteriores etapas para añadir las funcionalidades correspondientes a estas o ajustar las ya existentes.

- **updatevuln.class.php**: Contiene la clase **PluginNvdUpdatevuln,** que implementa la tarea principal que se encarga de consultar las diferentes versiones de los programas y sistemas operativos que se encuentran instalados en alguno de los equipos informáticos gestionados por GLPI, para después consultar la base de datos de NVD en busca de vulnerabilidades que afecten a estas versiones.

  Para obtener las instalaciones de los programas y sistemas operativos es necesario acceder a la base de datos del servidor en el que se encuentra la instancia de GLPI, lo que se puede llevar a empleando el generador de consultas a la base de datos [19] que provee GLPI.

  Concretamente se requiere consultar las siguientes tablas:

  - **glpi_softwareversions**: Contiene los identificadores, números de versión y sistema operativo y arquitectura objetivos de las distintas versiones de los programas gestionados por GLPI, así como referencias a sus correspondientes programas.
  - **glpi_items_softwareversions**: Contiene asociaciones entre dispositivos y versiones de programas gestionados por GLPI. Esta tabla puede interpretarse como las instalaciones individuales de cada una de las versiones de cada uno de los programas en cada uno de los dispositivos.
  - **glpi_operatingsystem[s, versions, kernels, kernelversions y servicepacks]**: Contienen el nombre, versión, núcleo, versión del núcleo y número de actualización de un sistema operativo. Esta información, que junta forma la versión de un sistema operativo, se encuentra referenciada desde la siguiente tabla.
  - **glpi_items_operatingsystems**: Contiene las asociaciones entre dispositivos y las versiones de los sistemas operativos instalados en estos. Esta tabla puede interpretarse como las instalaciones individuales de las versiones de cada uno de los sistemas operativos en cada uno de los dispositivos.
  - Otras tablas definidas por el plugin que se detallan en el apartado **4.4**.

- **connection.class.php**: Contiene la clase **PluginNvdConnection**, que implementa la configuración de una conexión a una API REST a través de http, concretamente el establecimiento de la url, parámetros y cabeceras de la consulta. Del mismo modo permite realizar la consulta y obtener la respuesta a la misma en formato JSON. Emplea la librería cURL [20] de PHP.
- **nvdconnection.php**: Contiene la clase **PluginNvdNvdconnection**, que hereda de **PluginNvdConnection** para definir la url base y posibles parámetros de la consulta a emplear para la API de la base de datos NVD.
- **cpe.class.php**: Contiene la clase **PluginNvdCpe**, que Facilita la creación de nombres con formato CPE para su uso en las consultas a la base de datos NVD. De igual forma es capaz de, dado un nombre en formato CPE, obtener del mismo cualquiera de sus atributos.

4.3. Tareas automáticas

La segunda funcionalidad que se ha decidido implementar para el plugin ha sido la capacidad de automatizar la tarea de consulta de vulnerabilidades a la base de datos NVD a partir de las versiones de programas y sistemas operativos de los equipos gestionados por GLPI obtenidos de la base de datos local. La capacidad de automatizar esta tarea y de programar la periodicidad de su ejecución permiten al administrador de GLPI adaptar de forma sencilla y eficiente la principal funcionalidad del plugin al contexto de trabajo de la entidad en el que se está empleando y la infraestructura informática que se está monitorizando.

Para lograr que esta tarea se ejecute de forma periódica y sin intervención explícita de ningún usuario, una vez ha sido definida, se ha hecho uso de las "*Automatic actions*" que ofrece el framework de desarrollador de GLPI [21]. Este mecanismo requiere de la definición de la tarea como función de una clase junto a una función de consulta de información en la misma clase y de un registro explícito de la tarea, normalmente durante la instalación del plugin, que especifique, entre otros parámetros, el nombre de la tarea, la clase de la que proviene, su periodicidad y su modo de activación.

4.3.1. Definición de la tarea

Para que GLPI detecte correctamente la tarea esta ha de definirse como una función estática pública con el prefijo "cron" antes del nombre de la terea como nombre de la función, es decir, si la tarea en cuestión va a llamarse **UpdateVulnTask**, se deberá crear la función **cronUpdateVulnTask** en la clase **PluginNvdUpdatevuln** del fichero **updatevuln.class.php**. Adicionalmente, para proveer al framework de tareas automáticas de GLPI con información acerca de la tarea definida se ha de crear en la misma clase la función **cronInfo**, también estática y pública, que devolverá en forma de vector campos de información como la descripción de la tarea.

4.3.2. Registro de la tarea

Una vez se ha definido la tarea en la clase correspondiente es necesario registrarla contra el framework de tareas automáticas de GLPI para que pueda ejecutarse sin más intervención humana. Para ello GLPI ofrece un método estático de la clase **CronTask**, definida en el *core* de GLPI, llamado **Register**. Este método toma como argumentos, en este orden, el nombre de la clase en la que se ha definido la tarea (**PluginNvdUpdatevuln**), el nombre de la tarea definida (**UpdateVulnTask**), la periodicidad de ejecución de la tarea en segundos y un vector de parámetros adicionales de entre los que destacar los siguientes:

- Comentario: Comentario sobre el funcionamiento de la tarea.
- Modo de arranque: Define el método por el que la tarea se ha de arrancar. Puede ser interno o externo.

Las tareas definidas con modo de arranque interno se ejecutan de una en una mediante el planificador interno de GLPI, que elige la tarea que lleve más tiempo preparada para ejecutarse. Este esquema de ejecución de tareas puede suponer que, en caso de que haya varias tareas con periodos cortos de ejecución definidas con modo de arranque interno, las tareas con periodos más largos y mismo modo de arranque se vean relegadas a esperar en una cola de ejecución en la que, posiblemente, múltiples ejecuciones de las tareas más rápidas se encuentren esperando a ser ejecutadas, desembocando así en un incumplimiento de los plazos de ejecución.

Para evitar esta situación existe el modo de arranque externo, que es el recomendado para las tareas definidas por usuarios y plugins de GLPI. En contraste con el modo interno, más apto para tareas del framework de GLPI, este modo depende de la activación externa de las tareas encoladas mediante un mecanismo del propio sistema en el que se está ejecutando la instancia de GLPI. Dependiendo del sistema operativo en el que se haya instalado se pueden emplear diferentes mecanismos para activar las tareas definidas en modo de arranque externo, pero los más comunes son crear un "*Cron job*" en distribuciones de tipo Linux o una tarea programada con el "*Programador de tareas*" en sistemas Windows. En cualquier caso, el mecanismo de arranque consistirá siempre en una llamada mediante el comando php al script **cron.php** localizado bajo la carpeta **front** en el directorio de instalación de GLPI (**glpi/front/cron.php**).

Las ventajas que presenta este método frente al arranque interno pasan, no sólo por la capacidad que tiene el administrador de escoger la frecuencia empleada en el método de arranque, sino por la capacidad de ejecutar múltiples tareas encoladas por cada activación externa, así como por la posibilidad de configurar el número máximo de tareas a ejecutar por activación. Es por estas ventajas por las que se ha optado por definir el modo de arranque de la tarea **UpdateVulnTask** como externo.

4.3.3. Funcionalidades adicionales de la tarea automática

A parte de automatizar las consultas de registros de vulnerabilidades a la base de datos NVD, se han definido una serie de subtareas que deberán de realizarse durante la ejecución de la tarea **UpdateVulnTask** para cumplir con las especificaciones definidas de la funcionalidad del plugin:

- Volcado de las vulnerabilidades encontradas en la base de datos local para permitir su visualización.
- Creación de asociaciones entre vulnerabilidades encontradas y versiones de software afectadas.
- Eliminación de vulnerabilidades de la base de datos que previamente se encontraran presentes en algún dispositivo gestionado por GLPI, pero ya no.

Para cumplir con estas especificaciones ha sido necesario modificar el esquema de la base de datos local de GLPI y adaptarlo a las necesidades del plugin, creando las tablas y relaciones pertinentes.

4.4. Persistencia en base de datos local

Como se ha mencionado anteriormente, ha sido necesario adaptar el esquema de la base de datos de GLPI para satisfacer las necesidades del plugin desarrollado, principalmente creando una serie de tablas y relaciones que soporten la persistencia de las vulnerabilidades encontradas en los dispositivos gestionados por GLPI, su relación con las diferentes versiones de software y sistemas operativos instalados en estos dispositivos, así como la adaptación de los nombres de programas y fabricantes gestionados por GLPI al estándar CPE o la configuración persistente del plugin.

Siguiendo la documentación de plugins de GLPI, concretamente el apartado de la base de datos [22], se nos indica que las tablas creadas han de estar precedidas siempre por el prefijo **glpi_plugin_** y el nombre del plugin, por lo que para el plugin desarrollado los nombres de todas sus tablas comenzarán por **glpi_plugin_nvd_** y se nos recomienda emplear claves numéricas autoincrementables para estas, lo que facilita la referenciación de estas desde otras tablas. La misma documentación nos ofrece métodos para la creación o destrucción de tablas en la base de datos a través de consultas configuradas en forma de cadenas de caracteres. Estos métodos se emplearán para la creación de las tablas durante la instalación del plugin y su destrucción durante la desinstalación del plugin.

Los procesos de creación y destrucción de tablas se han trasladado a los scripts php **install.php** y **uninstall.php**, ubicados en la carpeta **installation** del plugin. Las funciones que implementan estos procesos se llaman desde los "*hooks*" de instalación y desinstalación definidos en el fichero **hook.php**.

4.4.1. Tabla glpi_plugin_nvd_vulnerabilities

La primera tabla en ser creada en la base de datos está destinada a almacenar la información más relevante acerca de las vulnerabilidades encontradas en los dispositivos gestionados por GLPI para permitir su posterior visualización. De todos los campos contenidos en los registros de vulnerabilidades de la base de datos NVD se ha decidido almacenar los siguientes:

- Identificador numérico: Empleado como clave primaria.
- Identificador CVE de la vulnerabilidad: Con el objetivo de distinguir inequívocamente las vulnerabilidades encontradas.
- Descripciones: Descripciones disponibles de la vulnerabilidad en diferentes idiomas.
- Métricas de severidad: Valoraciones de severidad de CVSS que permitan de forma sencilla valorar la severidad de la vulnerabilidad hallada.
- Configuraciones afectadas: Nombres CPE de las configuraciones afectadas por la vulnerabilidad que permitan identificar el alcance de afectación de la misma en los equipos gestionados.

Debido a que un registro de una vulnerabilidad concreta puede contener descripciones en varios idiomas, así como múltiples configuraciones de software a las que afecta esta, se ha decidido extraer estos campos y almacenarlos en otras tablas con referencias a la vulnerabilidad a la que corresponden.

Adicionalmente se ha creado una restricción sobre el identificador CVE para que no pueda repetirse en diferentes entradas. El esquema de la tabla creada se muestra en la **Figura 4.**

Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
id	int(10)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT
cve_id	varchar(16)	utf8mb4_unicode_ci		No	*Ninguna*		
base_score	float			Sí	*NULL*		
exploitability_score	float			Sí	*NULL*		
impact_score	float			Sí	*NULL*		

Figura 4: Estructura de la tabla glpi_plugin_nvd_vulnerabilities.

4.4.2. Tabla glpi_plugin_nvd_vulnerability_descriptions

Se trata de la tabla destinada a almacenar las diferentes descripciones disponibles para una vulnerabilidad en concreto en los diferentes idiomas en los que se ha publicado en la base de datos NVD. Los campos que almacena esta tabla son los siguientes:

- Identificador numérico: Empleado como clave primaria.
- Identificador de vulnerabilidad: Identificador numérico que hace referencia a la clave primaria de una vulnerabilidad en la tabla glpi_plugin_nvd_vulnerabilities.
- Identificador de idioma: Identificador del idioma en el que se encuentra redactada la descripción. Este identificador, de dos caracteres de longitud, cumple con el estándar **ISO 639-1**.
- El contenido de la descripción en sí mismo tal y como se ha obtenido del registro de la base de datos NVD.

Se ha creado adicionalmente una restricción sobre los campos de identificador de vulnerabilidad e idioma para evitar que se repitan combinaciones de estos. El esquema de la tabla creada se muestra en la **Figura 5.**

Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
id	int(10)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT
vuln_id	int(10)		UNSIGNED	No	*Ninguna*		
language	char(2)	utf8mb4_unicode_ci		Sí	*NULL*		
description	varchar(8000)	utf8mb4_unicode_ci		Sí	*NULL*		

Figura 5: Estructura de la tabla glpi_plugin_nvd_vulnerability_descriptions

4.4.3. Tabla glpi_plugin_nvd_vulnerability_configurations

Esta tabla contiene, para cada vulnerabilidad registrada en la base de datos, una entrada por cada programa o sistema operativo al que afecta y que presenta alguna característica particular en su configuración, es decir, que la configuración concreta del software afectado presenta alguna restricción en alguno de los siguientes atributos del estándar CPE:

- *Update*: Actualización concreta del software en cuestión que manifiesta la vulnerabilidad.
- *Edition*: Edición concreta del software en cuestión que manifiesta la vulnerabilidad.
- *Target Software*: Sistema operativo concreto para el cual se ha diseñado el software en cuestión que manifiesta la vulnerabilidad.
- *Target Hardware*: Dispositivo hardware o arquitectura concreta en la cual ha de correr el software en cuestión que manifiesta la vulnerabilidad.

Además de esta información, almacenada en forma de listas de valores, las columnas que presentan los registros de esta tabla son las siguientes:

- Identificador numérico: Empleado como clave primaria.
- Identificador de vulnerabilidad: Identificador numérico que hace referencia a la clave primaria de una vulnerabilidad en la tabla glpi_plugin_nvd_vulnerabilities.
- Identificador del fabricante: Identificador del fabricante del software afectado por la vulnerabilidad en el estándar CPE.
- Identificador del producto: Identificador del producto en sí afectado por la vulnerabilidad en el estándar CPE.

Adicionalmente se ha creado una restricción sobre las columnas de los tres últimos identificadores definidos de forma que no se repitan combinaciones de estos. El esquema de la tabla creada se muestra en la **Figura 6.**

Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
id	int(10)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT
vuln_id	int(10)		UNSIGNED	No	*Ninguna*		
vendor_name	varchar(255)	utf8mb4_unicode_ci		Sí	*NULL*		
product_name	varchar(255)	utf8mb4_unicode_ci		Sí	*NULL*		
update	varchar(2047)	utf8mb4_unicode_ci		Sí	*NULL*		
edition	varchar(2047)	utf8mb4_unicode_ci		Sí	*NULL*		
target_sw	varchar(2047)	utf8mb4_unicode_ci		Sí	*NULL*		
target_hw	varchar(2047)	utf8mb4_unicode_ci		Sí	*NULL*		

Figura 6: Estructura de la tabla glpi_plugin_nvd_vulnerability_configurations.

4.4.4. Tabla glpi_plugin_nvd_vulnerable_versions

Esta tabla está destinada a almacenar las relaciones entre las vulnerabilidades encontradas en los dispositivos gestionados por GLPI, almacenadas en la tabla glpi_plugin_nvd_vulnerabilities, y las diferentes versiones de software a las que afectan, cuyos identificadores se almacenan en la tabla glpi_softwareversions. Las columnas que posee esta tabla son:

- Identificador numérico: Empleado como clave primaria.
- Identificador de vulnerabilidad: Identificador numérico que hace referencia a la clave primaria de una vulnerabilidad en la tabla glpi_plugin_nvd_vulnerabilities.
- Identificador de versión: Referencia numérica a una versión concreta de un programa almacenada en la tabla glpi_softwareversions.

Adicionalmente se ha creado una restricción sobre las columnas de los dos últimos identificadores definidos de forma que no se repitan combinaciones de estos. El esquema de la tabla creada se muestra en la **Figura 7**.

Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
id	int(10)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT
vuln_id	int(10)		UNSIGNED	No	*Ninguna*		
softwareversions_id	int(10)		UNSIGNED	No	*Ninguna*		

Figura 7: Estructura de la tabla glpi_plugin_nvd_vulnerable_software_versions.

4.4.5. Tabla glpi_plugin_nvd_vulnerable_system_versions

De forma análoga a la tabla anterior, esta tabla almacena las relaciones entre las vulnerabilidades encontradas en los dispositivos gestionados por GLPI, almacenadas en la tabla glpi_plugin_nvd_vulnerabilities, y las diferentes versiones de los sistemas operativos a los que afectan.

La información identificativa de cada versión de un sistema operativo puede llegar a encontrarse dispersa por hasta seis tablas diferentes gestionadas por GLPI, por lo que, en lugar de almacenar seis identificadores como referencia a cada una de las tablas, se ha escogido emplear una columna que almacene la configuración concreta del sistema operativo en cuestión empleando los campos *vendor*, *product* y *versión* del estándar CPE concatenados y separados por el carácter ‘:’. Así pues, a modo de ejemplo, un sistema operativo Microsoft Windows Server 2019 con la versión del *kernel* 10.0.17763.737 se almacenaría como “*microsoft:windows_server_2019:10.0.17763.737*”.

Este identificador habrá de generarse cada vez que se quiera insertar nuevos datos en la tabla, y será necesario dotar al plugin con la capacidad de, dada la información recogida por el agente de inventariado de GLPI acerca de un sistema operativo concreto, reconocer el tipo y versión de este y generar su correspondiente identificador. Esta funcionalidad se ha implementado en la clase **cpe.class.php**.

Así pues, teniendo en cuenta lo expuesto anteriormente, las columnas presentes en esta tabla serán las siguientes:

- Identificador numérico: Empleado como clave primaria.
- Identificador de vulnerabilidad: Identificador numérico que hace referencia a la clave primaria de una vulnerabilidad en la tabla glpi_plugin_nvd_vulnerabilities.
- Identificador de configuración: Cadena de caracteres que identifica de forma inequívoca, empleando los atributos *vendor*, *product* y *versión* del estándar CPE, una versión de un sistema operativo concreto.

Adicionalmente se ha creado una restricción sobre las columnas de los dos últimos identificadores definidos de forma que no se repitan combinaciones de estos. El esquema de la tabla creada se muestra en la **Figura 8**.

Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
id	int(10)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT
vuln_id	int(10)		UNSIGNED	No	*Ninguna*		
system_configuration	varchar(2047)	utf8mb4_unicode_ci		No	*Ninguna*		

Figura 8: Estructura de la tabla glpi_plugin_nvd_vulnerable_system_versions.

4.4.6. Tabla glpi_plugin_nvd_cpe_software_associations

Durante la fase de investigación del proyecto se indagó acerca de los métodos más adecuados para identificar las instalaciones de software en un equipo de forma inequívoca y, a partir de los identificadores obtenidos, consultar las bases de datos de vulnerabilidades en busca de registros que contuvieran configuraciones coincidentes con el software identificado. Se llegó a la conclusión de que el método más apropiado para realizar estas consultas sería obtener los nombres de las instalaciones en formato CPE, con los nombres de fabricante y producto y la versión del producto, y consultar la base de datos NVD filtrando mediante este parámetro.

Existe, sin embargo, una dificultad a la hora de obtener el nombre en formato CPE de un producto en concreto, y es que los nombres de fabricante y producto aceptados por el estándar CPE no coinciden, en la mayoría de los casos, con los nombres del fabricante y la aplicación en cuestión disponibles en el sistema. Como ejemplo ilustrativo se va a tomar el caso del conocido navegador de Google, Google Chrome.

Si se indaga en el sistema en busca de las propiedades del ejecutable del navegador se encontrará que el nombre del producto está definido como "*Google Chrome*", mientras que el fabricante del mismo está definido como "*Google LLC*". Si se consulta el diccionario oficial de nombres CPE, se podrá observar como su nombre en formato CPE es "*cpe:2.3:a:google:chrome*", es decir, no coinciden ni el nombre del fabricante ni el nombre del producto. Esto es debido a que el estándar de nomenclatura CPE no acepta cualquier nombre para sus campos, sino que estos han de encontrarse definidos en el diccionario oficial.

Para el caso concreto de los sistemas operativos, como se ha mencionado anteriormente, es viable, dentro del contexto de este proyecto, crear un mecanismo que realice una traducción exacta entre los nombres ofrecidos por el agente de inventariado de GLPI y los nombres aceptados por el estándar CPE. Esto se debe a que, en comparación con el número de aplicaciones diferentes en el mercado, el número de sistemas operativos existentes es mucho más reducido. Sin embargo, aunque la solución ideal pasaría por aplicar este mismo mecanismo a las aplicaciones software, su implementación se ha dejado fuera del alcance de este proyecto por lo costoso en tiempo que supondría.

Con esta dificultad en mente y con el objetivo de solucionar el problema que causa para realizar las consultas a la base de datos NVD en busca de vulnerabilidades para versiones de aplicaciones concretas, se ha creado una nueva tabla en la base de datos que almacenará, para cada software gestionado por GLPI del cual se deseen buscar vulnerabilidades, los nombres en el estándar CPE de su fabricante y el producto en sí mismo. La tarea automática de actualización de vulnerabilidades extraerá de esta tabla los nombres a emplear para la construcción del nombre CPE junto a la versión concreta del software en vez de consultar las tablas **glpi_softwares** y **glpi_manufacturers**, que utiliza GLPI para almacenar los nombres de las aplicaciones y fabricantes.

Esta aproximación, sin embargo, supondrá la necesidad de que, previa ejecución de la tarea, un usuario haya asignado a cada una de las aplicaciones de las cuales se desean buscar vulnerabilidades, los nombres de producto y fabricante en el estándar CPE de forma manual, los cuales se almacenarán en la nueva tabla cuyas columnas se detallan a continuación:

- Identificador numérico: Empleado como clave primaria.
- Identificador de software: Referencia a la clave primaria de la tabla glpi_softwares para identificar una aplicación.
- Nombre del fabricante: Nombre del fabricante del producto en el estándar CPE (*vendor*).
- Nombre del producto: Nombre del producto en sí mismo en el estándar CPE (*product*).

Adicionalmente se ha añadido una restricción sobre el identificador de software para evitar que se pueda asociar a una misma aplicación distintos nombres de fabricante y producto en el estándar CPE. El esquema de la tabla creada puede observarse en la **Figura 9**.

Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
id	int(10)		UNSIGNED	No	*Ninguna*		AUTO_INCREMENT
softwares_id	int(10)		UNSIGNED	No	*Ninguna*		
vendor_name	varchar(255)	utf8mb4_unicode_ci		Sí	*NULL*		
product_name	varchar(255)	utf8mb4_unicode_ci		Sí	*NULL*		

Figura 9: Estructura de la tabla glpi_plugin_nvd_cpe_software_associations.

4.4.7. Tabla glpi_plugin_nvd_config

Esta última tabla está destinada a almacenar parámetros de configuración empleados por el plugin y que pueden ser modificados por el usuario o administrador de GLPI. Actualmente esta tabla únicamente almacena la clave de la API de NVD empleada como método de autenticación durante las consultas a la base de datos NVD, sin embargo, en un futuro podría actualizarse para almacenar diferentes configuraciones para cada usuario definido en el framework de GLPI añadiendo como identificador una columna de referencia a la tabla que almacena los usuarios en la base de datos. El esquema de la tabla creada puede observarse en la **Figura 10**.

Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
api_key	varchar(63)	utf8mb4_unicode_ci		No	*Ninguna*		

Figura 10: Estructura de la tabla glpi_plugin_nvd_config.

4.4.8. Consistencia de la base de datos

Es altamente probable que, con el paso del tiempo, las vulnerabilidades y las referencias a las correspondientes versiones de programas y sistemas operativos vulnerables almacenadas en la base de datos de GLPI queden obsoletas con las actualizaciones aplicadas durante el ciclo de vida natural de estas instancias de software. Es por ello por lo que la tarea automática de actualización de vulnerabilidades deberá de ser capaz de detectar y eliminar de la base de datos las vulnerabilidades y relaciones con versiones de aplicaciones y sistemas operativos que ya no se encuentren presentes en los dispositivos gestionados por GLPI.

4.4.9. Clases que acceden a la base de datos

A lo largo del proyecto son múltiples los ficheros que se han creado para implementar las diferentes funcionalidades descritas en la fase de planificación, muchas de las cuales requieren acceder a la base de datos llevando a cabo consultas similares o incluso idénticas. Por esta razón se ha desarrollado la clase **PluginNvdDatabaseutils**, que se encuentra implementada en el fichero **databaseutils.class.php**. Los ficheros que implementan métodos que requieren de interacción con la base de datos y emplean esta clase, a parte de los anteriormente mencionadas (procesos de instalación y desinstalación del plugin y actualización de vulnerabilidades), son los siguientes:

- Clases dedicadas a generar vistas en GLPI: Estas clases, definidas en el apartado **4.5.1**, implementan la interfaz gráfica del plugin con la que el usuario puede interactuar. Se encuentran bajo en directorio **inc**.
- Métodos dedicados a procesar peticiones de formularios: Definidos en el mismo apartado, estos métodos responden a las interacciones entre el usuario y la interfaz del plugin que destinadas a modificar los contenidos de la base de datos. Se encuentran bajo en directorio **front**.

4.4.10. Esquema de las principales tablas de la base de datos

La **Figura 11** muestra el esquema de la base de datos de GLPI con las tablas creadas y las relaciones entre estas y las tablas nativas del framework de GLPI, de las cuales únicamente se muestran las columnas relevantes para la operativa del plugin.

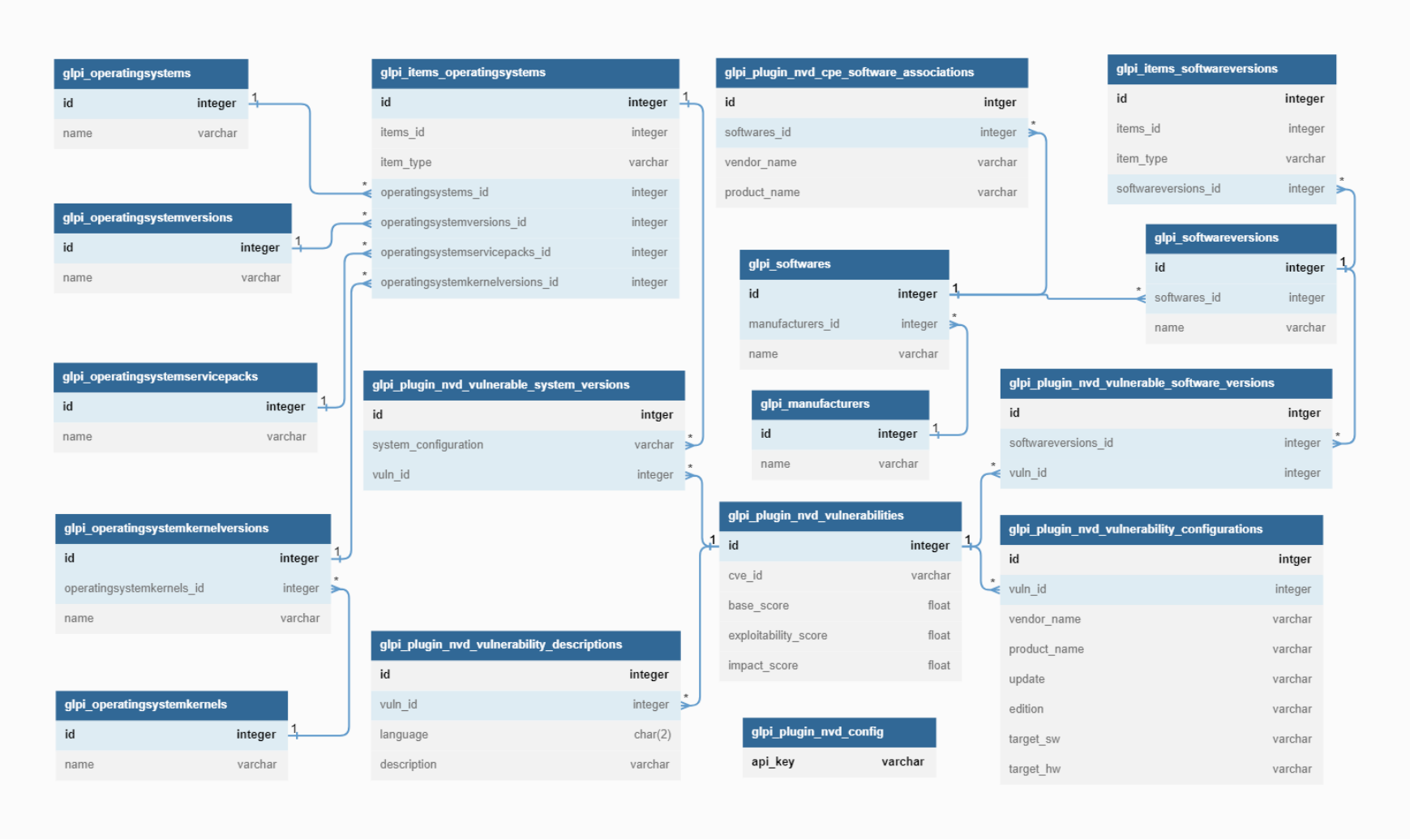

Figura 11: Esquema de las principales tablas de la base de datos de GLP. Generado con la herramienta dbdiagram.io **[23]**.

4.5. Inventariado de activos informáticos

Con el objetivo de probar las funcionalidades implementadas del plugin hasta el momento con inventarios recogidos en equipos reales se ha empleado el agente de inventariado de GLPI, concretamente la versión 1.4 del mismo [24]. Una vez instalado en el equipo, el agente puede ejecutarse de forma local a través de la línea de comandos o de forma remota mediante la solicitud de un servidor GLPI autorizado. Los procesos de instalación, configuración y ejecución del agente se encuentran documentados en la documentación dedicada del mismo [25].

Es posible configurar el agente para ejecutarse en línea en diferentes modos e importar de forma regular actualizaciones del inventariado de aplicaciones y redes al servidor GLPI, sin embargo, para el alcance del proyecto desarrollado, únicamente interesa contar con una muestra significativa de inventariados para realizar las pruebas de la funcionalidad implementada. Es por esto por lo que, en lugar de montar una compleja configuración de inventariado automático sincronizada con la instancia del servidor GLPI, se ha optado por emplear los scripts que ofrece el agente instalado, en forma de ficheros de lotes (.bat) en instalaciones en sistemas Windows o scripts ejecutables de perl en instalaciones Linux o MacOS, para generar un informe del inventariado de varias máquinas en formato JSON. Una vez generado el informe, este puede ser importado y asimilado desde la interfaz gráfica de GLPI, creándose (o modificándose) las entradas pertinentes en la base de datos de GLPI para reflejar en los activos gestionados el estado del inventario de los dispositivos cuyo archivo se ha importado.

Figura 12: Importación del archivo JSON de inventariado en GLPI.

Se puede observar directamente a través de la base de datos o mediante la interfaz gráfica de GLPI las entradas creadas para los diferentes dispositivos cuyos inventarios se han importado, así como los registros de los diferentes programas instalados en estos y sus correspondientes versiones. Es importante cotejar los resultados obtenidos del inventariado y compararlos con los formatos del estándar CPE para comprender la necesidad de realizar una traducción de los nombres de los fabricantes y productos de forma que coincidan con los presentes en el diccionario oficial. Otros parámetros obtenidos del inventariado, como las actualizaciones, ediciones o hardware específico asociados a una versión de un programa en concreto también necesitarían traducción al estándar CPE en caso de emplearse en las consultas.

4.5.1. Comparación entre resultados del inventariado y estándar CPE

Tal y como se ha adelantado en apartados anteriores, tras llevar a cabo la comparación entre los nombres de productos, fabricantes y versiones de programas y sistemas operativos recogidos en los resultados del inventariado llevado a cabo por el agente de GLPI se ha constatado la necesidad de crear un mecanismo de traducción entre ambos. Dicho mecanismo se ha implementado, a modo de ejemplo, para un número reducido de sistemas operativos, sin embargo, se ha optado por una aproximación diferente para el caso de las aplicaciones software.

La solución escogida ha sido la de crear una vista en la aplicación que permita a un usuario asignar de manera manual los nombres de fabricante y producto en el estándar CPE a cada aplicación en concreto. Por el contrario, en la gran mayoría de los casos, se ha observado que la versión de los programas recogida en los inventariados de GLPI cumple con el estándar CPE, por lo que no es necesario establecer manualmente este valor, lo que facilita considerablemente la tarea del usuario de GLPI. La creación de la vista empleada para la asignación de nombres CPE se detalla en el apartado **4.6**.

El mecanismo de traducción al estándar CPE implementado para las versiones de los sistemas operativos de los equipos gestionados por GLPI, que puede encontrarse en el fichero **cpe.class.php**, permite identificar diferentes sistemas operativos de tres familias distintas:

- Windows: Se ha implementado la detección de las versiones de los sistemas operativos Windows para ordenadores personales (Windows nt, 95, 98, 2000, xp, vista, 7, 8, 10 y 11) y para servidores (Windows Server 1709, 1803, 2000, 2003, 2004, 2008, 2012, 2016, 2019 y 2022).
- Linux: De los posibles sistemas operativos de la familia Linux se ha implementado la identificación de las versiones de Debian, Ubuntu y Redhat, por ser los más usados.
- MacOS: Por último, se ha implementado la identificación de las versiones de sistemas operativos de ordenadores apple.

Durante la implementación del mecanismo de traducción se comprobó que la versión 1.4 del agente de GLPI no era capaz de obtener el número de compilación completo de un sistema operativo Windows. Esta información se encuentra almacenada en el Registro de Windows en dos claves de la ruta "**HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion**", concretamente las claves "**CurrentBuildNumber**" y "**UBR**", siendo el número contenido en esta última el faltante. Dado que esta información es necesaria para obtener la versión completa de un sistema operativo Windows, se contactó a los programadores del agente de inventariado para proponer añadir la capacidad de obtenerla al agente, no obstante, se descubrió que esta funcionalidad se encuentra implementada en las versiones no finales de la futura versión 1.5 del agente de inventariado [26]. Así pues, se ha empleado la última versión disponible que implementa esta funcionalidad (**v1.5-git73ef762f** [27]) para generar los inventariados de equipos Windows y permitir así identificar correctamente las configuraciones de estos sistemas y poder emplearlas en las búsquedas de vulnerabilidades conocidas.

4.6. Creación de vistas y generación de alertas

Una vez se han definido una serie de funcionalidades que requieren de la interacción del usuario con la base de datos, es conveniente proporcionar a este una interfaz gráfica a través de la cual realizar las operaciones oportunas. De forma similar es recomendable crear vistas gráficas dentro de la aplicación para facilitar la visualización de la información de las diferentes vulnerabilidades que, empleando las versiones de los programas y sistemas operativos instalados en los diferentes equipos gestionados por GLPI, el plugin ha obtenido de la base de datos NVD. Con estas necesidades en mente, se han creado una serie de vistas en diferentes puntos de la interfaz gráfica del plugin empleando con HTML para la estructura de las páginas, CSS para mejorar la presentación de los elementos en estas y Javascript para aportarles dinamismo.

El plugin emplea una única hoja de estilo en formato de fichero CSS que puede encontrarse en la ruta **css/nvd.css**. De igual manera todas las funciones implementadas en javascript se hallan en el fichero **js/nvd.js**. Estos recursos se añaden de forma global a cada una de las páginas del plugin desde el fichero **setup.php**. Desde este mismo fichero se ha de indicar el punto exacto de la interfaz estándar de GLPI en el que la interfaz implementada por el plugin en una clase php en particular se añade.

4.6.1. Configuración del plugin

La primera vista en ser implementada responde a la necesidad de permitir a un usuario de la herramienta GLPI configurar la clave de la API que se utilizará en las consultas a la base de datos NVD para buscar vulnerabilidades. Esta clave, que se genera a través de una petición desde la página oficial del NIST [28], es uno de los parámetros requeridos por la API de este servicio para la consulta de vulnerabilidades, por lo que, en caso de no haberse configurado, la tarea de actualización automática de vulnerabilidades no podrá llevar a cabo su función.

La vista en sí se encuentra definida en el fichero **config.class.php** y presenta al usuario un formulario simple con el que actualizar el valor de la clave almacenado en la base de datos, mientras que el código encargado de la respuesta a la petición del formulario está definido en el fichero **config.form.php**, que se encuentra en el directorio **front**. Las peticiones de actualización del valor por parte de un usuario resultarán en la interacción con la base de datos de GLPI actualizando (o creando en caso de que no exista todavía) el valor de la columna **api_key** de la tabla **glpi_plugin_nvd_config**.

El punto concreto de la interfaz base de GLPI en el que se ha decidido localizar esta página de configuración es en el propio menú de configuración general de la herramienta y se le ha dado a la sección particular del plugin en este menú el nombre **NVD Plugin** (Configuración > General > NVD Plugin).

La **Figura 13** muestra esta vista en el punto de la interfaz del plugin indicado pudiéndose observar la caja de texto en la que introducir el valor concreto de la clave y el botón empleado para actualizarlo.

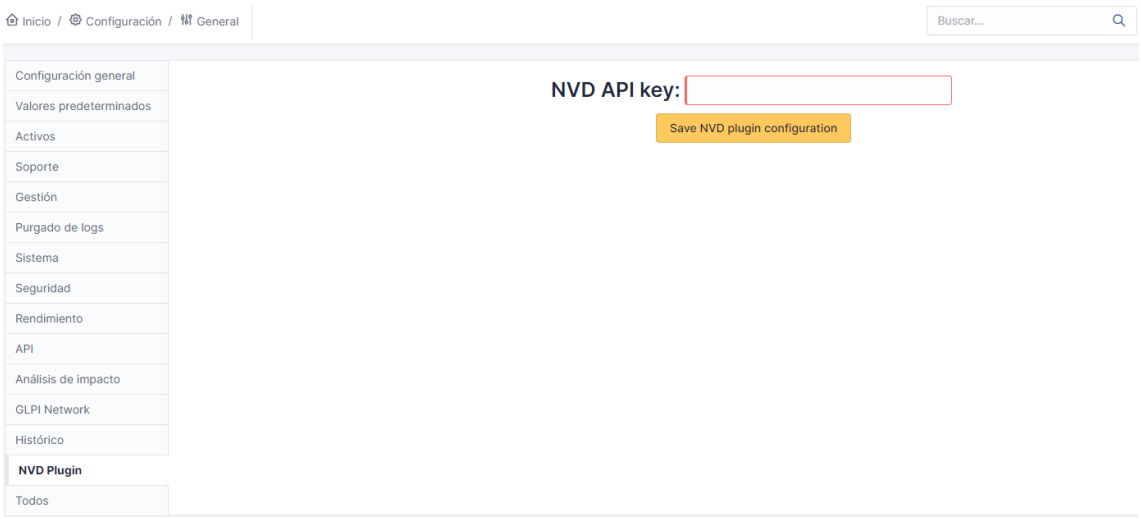

Figura 13: Vista de configuración del plugin NVD.

4.6.2. Asociación entre programa y nombres de fabricante y producto CPE

La otra funcionalidad implementada que requiere la interacción del usuario de GLPI con alguna de las tablas definidas por el plugin en la base de datos se corresponde con la necesidad, explicada en apartados anteriores, de almacenar asociaciones entre programas presentes en el inventario de GLPI y sus correspondientes nombres del fabricante y producto en el estándar CPE. Debido a que esta tarea puede llegar a ser relativamente ardua para un usuario, se ha planteado diseñar una interfaz que le facilite la interacción y la búsqueda de los nombres deseados entre aquellos disponibles y aprobados en el diccionario oficial del estándar CPE. Las características de la interfaz implementada destinadas a facilitar esta tarea al usuario de GLPI son las siguientes:

- Mostrar, en dos desplegables separados, listas de nombres de fabricantes y productos del estándar CPE de forma que, al seleccionar un fabricante del primer desplegable, se incluyen en el segundo desplegable únicamente los nombres de productos de tal fabricante. Al cambiar el fabricante seleccionado cambian los productos disponibles, y únicamente habiendo seleccionado ambos valores puede actualizarse la asociación en la base de datos.
- Incluir filtros de texto para los elementos de ambos desplegables, lo que permite al usuario, a través de dos botones de **aplicar** y **limpiar**, filtrar entre los múltiples valores disponibles en los desplegables dejando únicamente los que contengan el texto introducido en el filtro. De la misma manera puede deshacerse el filtrado o cambiarse por un nuevo valor.
- Mostrar sugerencias para los nombres del fabricante y producto basadas en los valores recogidos por el agente de inventariado de GLPI que, si bien no son exactamente los mismos que en el estándar CPE, contienen normalmente la mayor parte de las palabras que forman el nombre en este estándar. Estos valores, asociados a cada programa gestionado por GLPI, se almacenan en las tablas **glpi_softwares** y **glpi_manufacturers**.

- Permitir aplicar las sugerencias mostradas a los filtros de los desplegables de fabricantes y productos. Mostrando las sugerencias en forma de desplegables separados para los nombres de fabricante y producto, al seleccionar una de estas sugerencias el valor seleccionado pasa a ocupar el lugar del filtro, permitiendo aplicarlo de la misma forma que un filtro escrito a mano por un usuario.
- Mostrar, en caso de haberla, la asociación actual entre el programa seleccionado y los nombres de fabricante y producto en el estándar CPE almacenados en la base de datos. En caso contrario, mostrar un texto que indique que el programa seleccionado no cuenta con ninguna asociación establecida.

La vista en sí se encuentra definida en el fichero **softwarecpe.class.php** y presenta al usuario un formulario con el que actualizar el valor de los nombres de fabricante y producto en el estándar CPE asociados a un programa en concreto. Este formulario se compone de dos desplegables, para los nombres de fabricante y producto, y un botón con el que confirmar la asignación. Adicionalmente la interfaz presenta la asignación actual y un sistema de filtrado con el que reducir las opciones mostradas en los desplegables. Por otro lado, el código encargado de la respuesta a la petición del formulario está definido en el fichero **softwarecpe.form.php**, que se encuentra en el directorio **front**. Las peticiones de actualización de los valores por parte de un usuario resultarán en la interacción con la base de datos de GLPI actualizando (o creando en caso de que no exista todavía) el registro correspondiente al programa seleccionado en la tabla **glpi_plugin_nvd_cpe_software_associations**.

El punto concreto de la interfaz base de GLPI en el que se ha decidido localizar esta vista es en el apartado de *Software* dentro del menú de activos de la herramienta y se le ha dado a la sección particular del plugin en este menú el nombre **CPE Association** (Activos > Software > CPE Association). En la **Figura 14** se puede observar la vista para el programa Microsoft Teams, mientras que en la **Figura 15** se muestra la vista una vez realizada la asignación.

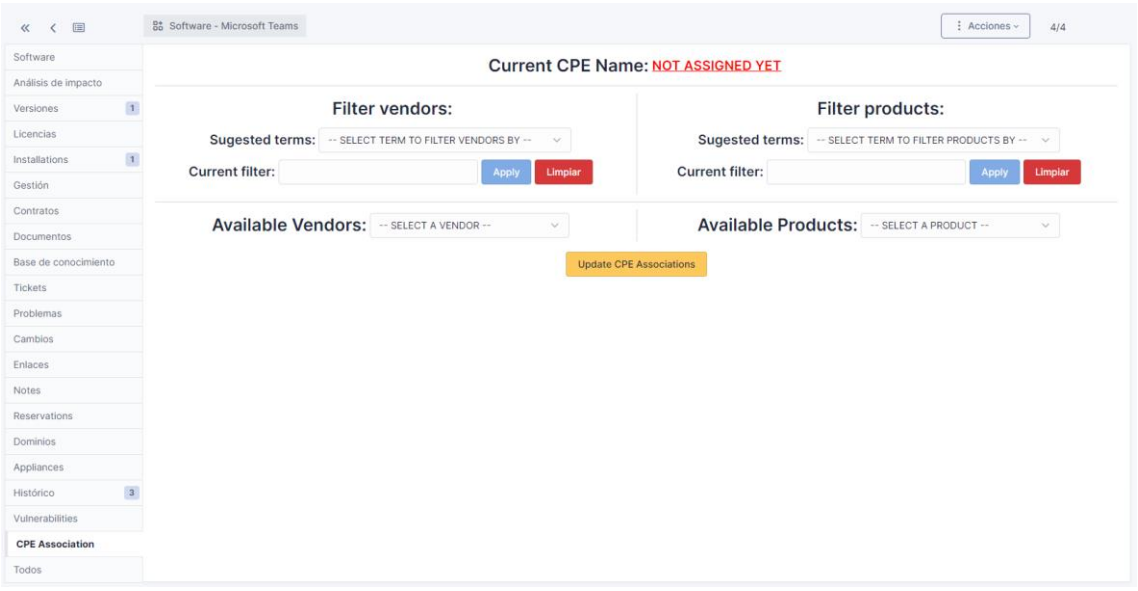

Figura 14: Vista de asignación de nombres de fabricante y producto CPE para un programa.

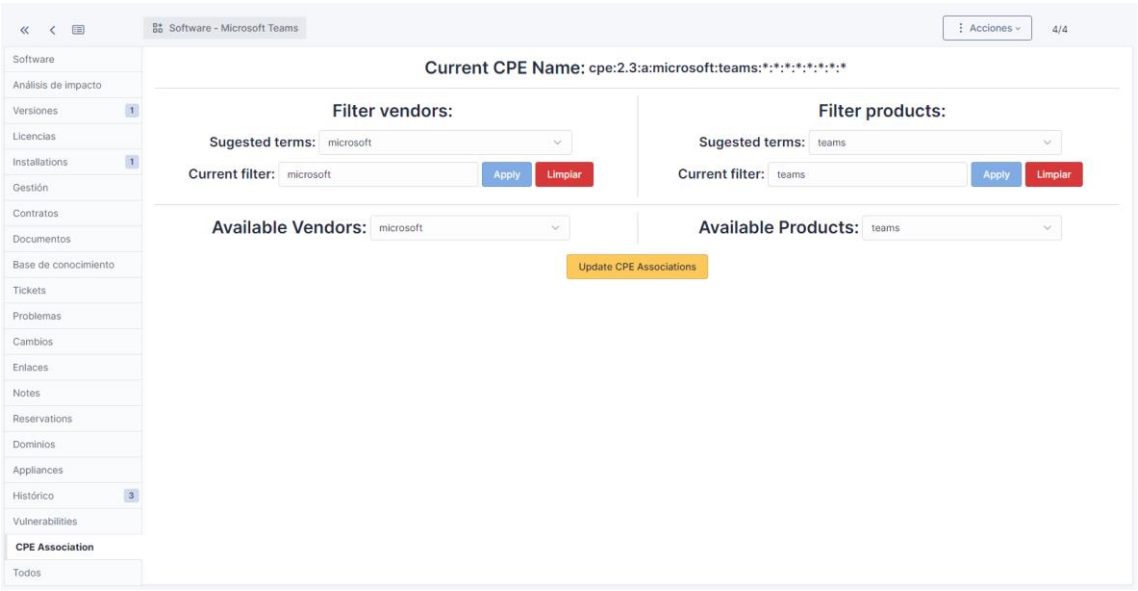

Figura 15: Vista tras la asignación de nombres CPE realizada.

4.6.3. Visualización de vulnerabilidades

Las vistas restantes creadas para el plugin pueden agruparse debido a la funcionalidad que comparten, que se trata de mostrar, de manera estructurada y ordenada, las diferentes vulnerabilidades que se han encontrado y almacenado en la base de datos que se encuentran relacionadas con las distintas versiones de los programas y sistemas operativos instalados en el conjunto de los equipos gestionados por GLPI. Se han definido en total tres vistas diferentes en distintos puntos de la interfaz base de GLPI:

- **Vista general**: Visualización del conjunto completo de vulnerabilidades recogidas en la base de datos de GLPI.
- **Vista de dispositivo**: Visualización del conjunto de vulnerabilidades que afectan a un dispositivo en concreto.
- **Vista de software**: Visualización del conjunto de vulnerabilidades que afectan a un programa en concreto.

Estas vistas, que se encuentran definidas en el fichero **vuln.class.php**, muestran al usuario una tabla conteniendo la información más relevante acerca de las vulnerabilidades recogidas de la base de datos NVD. Dependiendo de cada vista en concreto la información puede variar, sin embargo, la siguiente información se muestra en cada una de las tres vistas definidas:

- **Severidad / Puntuación**: Estas dos columnas indican la peligrosidad de una vulnerabilidad mediante una categorización (Crítica, Alta, Media, Baja o Nula) y nota numérica (0.0-10.0) respectivamente.
- **Identificador CVE**: Identificador de la vulnerabilidad asignado por el proyecto CVE de MITRE. A su vez es un enlace al registro de la vulnerabilidad en la base de datos NVD.
- **Descripción**: Breve descripción de la vulnerabilidad, en que consiste, afectación y método y consecuencias de explotación.

4.6.3.1. Visualización conjunta de vulnerabilidades

Esta vista permite visualizar la totalidad de las vulnerabilidades registradas en la base de datos de GLPI. El punto concreto de la interfaz de GLPI en el que se ha definido es la página de inicio, pudiéndose acceder a esta nueva sección a través de la pestaña con el nombre **Vulnerabilities**. Esta pestaña, adicionalmente, muestra al lado del nombre el número total de vulnerabilidades distintas registradas en la base de datos, dando al usuario una primera referencia de la información que encontrará una vez acceda al contenido de la vista.

Una vez se accede a la vista definida, se puede observar una división del contenido de en dos secciones diferenciadas mediante sendas pestañas:

- **Vulnerabilidades de programas**: En esta sección se muestran las todas vulnerabilidades relacionadas con versiones de algún programa en concreto. La pestaña tiene el nombre **Software Vulnerabilities** y nuevamente va acompañada del número de vulnerabilidades que se muestran al acceder a esta.
- **Vulnerabilidades de sistemas operativos**: En esta sección se muestran las todas vulnerabilidades relacionadas con versiones de algún sistema operativo en concreto. La pestaña tiene el nombre **OS Vulnerabilities** y, de la misma forma, presenta el número de vulnerabilidades que contiene.

Ambas secciones presentan las vulnerabilidades en formato tabular con la información ya mencionada. Adicionalmente, por cada vulnerabilidad, se muestran dos columnas:

- **Devices (Dispositivos)**: Contiene una lista de los dispositivos gestionados por GLPI en los que se da tal vulnerabilidad. Cada nombre mostrado es un enlace a la vista del dispositivo en la aplicación, lo que facilita al usuario la navegabilidad y le permite rápidamente acceder a la vista de vulnerabilidades para un dispositivo en concreto.
- **(Aviso)**: Esta columna contiene un aviso en caso de que alguna de las configuraciones para las que se da la vulnerabilidad en cuestión presente alguna restricción en alguno de los atributos del estándar CPE. El aviso se muestra en forma de marca de exclamación roja (❗) que, al ser seleccionada muestra un aviso por pantalla.

El aviso en cuestión se añade debido a que, a la hora de realizar las consultas de vulnerabilidades a la base de datos NVD, únicamente se emplean los campos “*vendor*”, “*product*” y “*version*” para identificar una instalación concreta de un programa o sistema operativo, dejando el resto de campos sin especificar. Esta decisión se ha tomado al observar que las instalaciones recabadas por el agente de inventariado, en muchas ocasiones, no incluyen esta información y, cuando lo hacen, no concuerda con el estándar CPE, por lo que nuevamente habría que implementar un mecanismo de traducción para cada uno de estos campos. En su lugar se ha decidido almacenar las configuraciones afectadas por las vulnerabilidades recabadas que presentaran alguna restricción en estos campos y mostrar un aviso en caso de que exista, para una vulnerabilidad en concreto, alguna configuración de estas características.

Cabe destacar que el número de vulnerabilidades mostrado en la pestaña **Vulnerabilities** puede ser menor que la suma de los números mostrados en las pestañas de las secciones **Software Vulnerabilities** y **OS Vulnerabilities**. Esto se debe a que puede darse el caso de que existan vulnerabilidades que afecten al mismo tiempo a configuraciones de algún programa y algún sistema operativo, por lo que se mostrarán en ambas pestañas, teniéndose en cuenta para sus recuentos particulares, pero únicamente se tendrán en cuenta una vez para el recuento total de vulnerabilidades.

En la **Figura 16** puede observarse la vista general de vulnerabilidades en la sección de vulnerabilidades de programas, mientras que la **Figura 17** muestra la misma vista en la sección de vulnerabilidades de sistemas operativos. Por último, en la **Figura 18** puede verse la alerta mostrada al hacer click sobre un aviso de alguna de las vulnerabilidades marcadas con la exclamación roja.

Severity	Score	CVE-ID		Descripción	Devices
CRITICAL	9.8	CVE-2022-28738		Se ha encontrado una doble liberación en el compilador de Regexp en Ruby versiones 3.x anteriores a 3.0.4 y versiones 3.1.x anteriores a 3.1.2. Si una víctima intenta crear un Regexp a partir de una entrada de usuario no confiable, un atacante puede ser capaz de escribir en ubicaciones de memoria no esperadas	debian
HIGH	8.8	CVE-2021-33621		The cgi gem before 0.1.0.2, 0.2.x before 0.2.2, and 0.3.x before 0.3.5 for Ruby allows HTTP response splitting. This is relevant to applications that use untrusted user input either to generate an HTTP response or to create a CGI::Cookie object.	debian
HIGH	8.1	CVE-2019-16255		Ruby versiones hasta 2.4.7, versiones 2.5.x hasta 2.5.6 y versiones 2.6.x hasta 2.6.4, permite una inyección de código si el primer argumento (también conocido como el argumento "command") para Shell#[] o Shell#test en la biblioteca lib/shell.rb es un dato no seguro. Un atacante puede explotar esto para llamar a un método de Ruby arbitrario.	Ubuntu
HIGH	7.5	CVE-2022-28739		Se presenta una lectura excesiva del búfer en Ruby versiones anteriores a 2.6.10, 2.7.x versiones anteriores a 2.7.6, 3.x versiones anteriores a 3.0.4 y 3.1.x versiones anteriores a 3.1.2. Es producida en la conversión String-to-Float, incluyendo Kernel#Float y String#to_f	debian,Ubuntu
HIGH	7.5	CVE-2021-41819		CGI::Cookie parse en Ruby versiones hasta 2.6.8, maneja inapropiadamente los prefijos de seguridad en los nombres de las cookies. Esto también afecta a CGI gem versiones hasta 0.3.0 para Ruby.	debian,Ubuntu
HIGH	7.5	CVE-2021-28966		En Ruby versiones hasta 3.0 en Windows, un atacante remoto puede enviar una ruta diseñada cuando una aplicación web maneja un parámetro con TmpDir	Ubuntu
HIGH	7.5	CVE-2021-28965		El REXML gem versiones anteriores a 3.2.5 en Ruby versiones anteriores a 2.6.7, versiones 2.7.x anteriores a 2.7.3 y versiones 3.x anteriores a 3.0.1, no aborda apropiadamente los problemas round-trip de XML. Puede ser producido un documento incorrecto después de analizarlo y serializarlo	Ubuntu
HIGH	7.5	CVE-2020-25613		Se detectó un problema en Ruby versiones hasta 2.5.8, versiones 2.6.x hasta 2.6.6 y versiones 2.7.x hasta 2.7.1. WEBrick, un simple servidor HTTP integrado con Ruby, no había comprobado rigurosamente el valor del encabezado transfer-encoding. Un atacante puede explotar potencialmente este problema para omitir un proxy inverso (que también presenta una comprobación de encabezado deficiente), que puede conllevar a un ataque de Trafico Inapropiado de Peticiones HTTP	Ubuntu
HIGH	7.5	CVE-2020-5247		En Puma (RubyGem) anterior a la versión 4.3.2 y anterior a la versión 3.12.3, si una aplicación que usa Puma permite la entrada no segura en un encabezado de respuesta, un atacante puede usar caracteres de nueva línea (es decir, 'CR',' LF' o' / r', ' / n') para finalizar el encabezado e inyectar contenido malicioso, como encabezados adicionales o un cuerpo de respuesta completamente nuevo. Esta vulnerabilidad se conoce como división de respuesta HTTP. Si bien no es un ataque en sí mismo, la división de la respuesta es un vector para varios otros ataques, como las secuencias de cross-site scripting (XSS). Esto está relacionado con CVE-2019-16254, que corrigió esta vulnerabilidad para el servidor web WEBrick Ruby. Esto se ha solucionado en las versiones 4.3.2 y 3.12.3 verificando todos los encabezados para ver los finales de línea y rechazando los encabezados con esos caracteres.	Ubuntu

Figura 16: Vista general de vulnerabilidades de programas.

Severity	Score	CVE-ID		Descripción	Devices
CRITICAL	10	CVE-2022-0543		Se ha detectado que redis, una base de datos persistente de valores clave, debido a un problema de empaquetado, es propenso a un escape del sandbox de Lua (específico de Debian), que podría resultar en una ejecución de código remota	debian
CRITICAL	10	CVE-2021-38503		Las reglas del sandbox de iframe no se aplicaban correctamente a las hojas de estilo XSLT, permitiendo a un iframe omitir restricciones como la ejecución de scripts o la navegación por el marco de nivel superior. Esta vulnerabilidad afecta a Firefox versiones anteriores a 94, Thunderbird versiones anteriores a 91.3 y Firefox ESR versiones anteriores a 91.3	debian
CRITICAL	10	CVE-2021-44228		Las características JNDI de Apache Log4j2 2.0-beta9 hasta 2.15.0 (excluyendo las versiones de seguridad 2.12.2, 2.12.3 y 2.3.1) utilizadas en la configuración, los mensajes de registro y los parámetros no protegen contra LDAP controlado por un atacante y otros puntos finales relacionados con JNDI. Un atacante que pueda controlar los mensajes de registro o los parámetros de los mensajes de registro puede ejecutar código arbitrario cargado desde servidores LDAP cuando la sustitución de la búsqueda de mensajes está habilitada. A partir de la versión 2.15.0 de log4j, este comportamiento ha sido deshabilitado por defecto. A partir de la versión 2.16.0 (junto con las versiones 2.12.2, 2.12.3 y 2.3.1), esta funcionalidad se ha eliminado por completo. Tenga en cuenta que esta vulnerabilidad es específica de log4j-core y no afecta a log4net, log4cxx u otros proyectos de Apache Logging Services	debian
CRITICAL	10	CVE-1999-0590		A system does not present an appropriate legal message or warning to a user who is accessing it.	MacBook Air de Jesús, MacBook Pro de Sergio (2)
CRITICAL	9.9	CVE-2021-21345		XStream es una biblioteca de Java para serializar objetos a XML y viceversa. En XStream anterior a la versión 1.4.16, se presenta una vulnerabilidad que puede permitir a un atacante remoto que tenga suficientes derechos ejecutar comandos del host solo manipulando el flujo de entrada procesado. Ningún usuario está afectado, si siguió la recomendación de configurar el framework de seguridad de XStream con una lista blanca limitada a los tipos mínimos requeridos. Si confía en la lista negra predeterminada de XStream del Framework de Seguridad, tendrá que usar al menos la versión 1.4.16	debian
CRITICAL	9.8	CVE-2021-44538		La función olm_session_describe en Matrix libolm versiones anteriores a 3.2.7, es vulnerable a un desbordamiento de búfer. El objeto Olm session representa un canal criptográfico entre dos partes. Por lo tanto, su estado es parcialmente controlable por la parte remota del canal. Los atacantes pueden construir una secuencia de mensajes manipulada para manipular el estado de la sesión del receptor de tal manera que, para algunos tamaños de búfer, se produzca un desbordamiento de búfer en una llamada a olm_session_describe. Además, los tamaños de búfer seguros no estaban documentados. El contenido del desbordamiento es parcialmente controlable por el atacante y se limita a espacios y dígitos ASCII. Los productos afectados conocidos son Element Web y SchildiChat Web	debian
CRITICAL	9.8	CVE-2021-44790		Un cuerpo de petición cuidadosamente diseñado puede causar un desbordamiento de búfer en el analizador multiparte mod_lua (r:parsebody() llamado desde scripts Lua). El equipo de Apache httpd no presenta constancia de que se presente una explotación para esta vulnerabilidad, aunque podría ser posible diseñar uno. Este problema afecta a Apache HTTP Server versiones 2.4.51 y anteriores	debian

Figura 17: Vista general de vulnerabilidades de sistemas operativos.

Figura 18: Alerta mostrada al hacer click en un aviso.

4.6.3.2. Visualización de vulnerabilidades por equipo

Esta vista permite visualizar el subconjunto de las vulnerabilidades registradas en la base de datos de GLPI que se encuentran presentes en uno de los equipos gestionados por la herramienta en concreto. Esta vista puede ser accedida desde dos puntos del menú de activos de la herramienta, habiéndose implementado tanto para computadores como para teléfonos y pudiéndose acceder a esta nueva sección a través de la pestaña con el nombre **Vulnerabilities** (Activos > [Computadores/Teléfonos] > Vulnerabilities). Esta pestaña, al igual que sucedía en el caso de la vista general, muestra al lado del nombre el número total de vulnerabilidades distintas que afectan, esta vez, al dispositivo seleccionado, dando al usuario una primera referencia de la información que encontrará una vez acceda al contenido de la vista.

Siguiendo con las similitudes con la anterior vista definida, esta nueva vista también se divide en dos secciones que muestran las vulnerabilidades relacionadas con versiones de programas y sistemas operativos respectivamente en formato tabular. En este caso, sin embargo, las columnas de las secciones no coinciden en su totalidad con las mostradas en las secciones de la vista general:

- **Vulnerabilidades de programas**: En esta sección la columna **Devices** se sustituye por la columna **Programs**. La nueva columna contiene, en lugar de una lista de dispositivos afectados por la vulnerabilidad, una lista de programas, que se encuentran instalados en el dispositivo seleccionado, afectados por la vulnerabilidad en cuestión. Nuevamente cada uno de los nombres mostrados actúa como enlace a la vista del programa correspondiente en la aplicación, facilitando al usuario la navegabilidad y permitiéndole acceder de forma rápida a la vista de vulnerabilidades del programa seleccionado.
- **Vulnerabilidades de sistemas operativos**: En esta sección la columna **Devices** desaparece por completo. Esto se debe a que, para un dispositivo en concreto, únicamente puede existir una configuración de sistema operativo que se encuentre instalada, por lo que no tendría sentido seguir mostrando una lista de configuraciones afectadas, como sí se hace en la sección de vulnerabilidades de programas. Adicionalmente cabe mencionar que, en esta sección, en la columna de aviso únicamente se marcarán aquellas vulnerabilidades para las cuales exista una configuración concreta del sistema operativo del dispositivo seleccionado que presente alguna restricción en alguno de los atributos del estándar CPE.

De forma similar a la vista general de vulnerabilidades, el número de vulnerabilidades mostradas en la pestaña principal podrá ser menor que la suma de los mostrados en las pestañas de las secciones de vulnerabilidades de programas (**Figura 19**) y sistema operativo (**Figura 20**) en caso de que alguna vulnerabilidad en particular se comparta entre estas secciones. Por otro lado, cabe resaltar que las alertas generadas para las vulnerabilidades relacionadas con programas serán similares a la mostrada en la **Figura 18**, mientras que en caso de generarse para el sistema operativo del dispositivo mostrarán los campos CPE y sus restricciones como puede observarse en la **Figura 21**.

Figura 19: Vista de vulnerabilidades de programas instalados en un dispositivo.

Figura 20: Vista de vulnerabilidades del sistema operativo de un dispositivo.

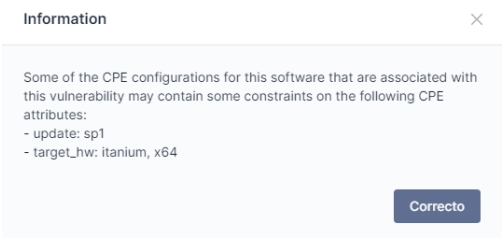

Figura 21: Alerta de vulnerabilidad con restricciones para un sistema operativo concreto.

4.6.3.3. Visualización de vulnerabilidades por programa

La última vista definida permite la visualización de las vulnerabilidades recogidas en la base de datos de GLPI relacionadas con un programa en concreto. El punto concreto de la interfaz de GLPI en el que se ha localizado es en el apartado de *Software* dentro del menú de activos de la herramienta, pudiéndose acceder a esta nueva sección a través de la pestaña con el nombre **Vulnerabilities** (Activos > Software > Vulnerabilities). Nuevamente el nombre de la pestaña va acompañado del número total de vulnerabilidades que afectan, en este caso, al programa seleccionado, dando al usuario una primera referencia de la información que encontrará una vez acceda al contenido de la vista.

En este caso, sin embargo, la vista no se divide en dos secciones diferentes, puesto que únicamente se muestran vulnerabilidades del programa. En esta única sección, mostrada en la **Figura 22**, la columna que anteriormente mostraba los dispositivos, en la vista general, o los programas, en la vista de dispositivo, en los cuales estaba presente la vulnerabilidad se ha sustituido por una columna, con el nombre **Versions**, que lista las diferentes versiones del programa seleccionado que presentan tal vulnerabilidad (Cada nombre individual es nuevamente un enlace a la correspondiente vista de versiones de programas en GLPI). Por último, se ha de mencionar que las alertas generadas para las vulnerabilidades mostrarán los campos CPE y sus restricciones, de forma similar a lo mostrado en la **Figura 21**, como puede observarse en la **Figura 23**.

Figura 22: : Vista de vulnerabilidades de las versiones de un programa.

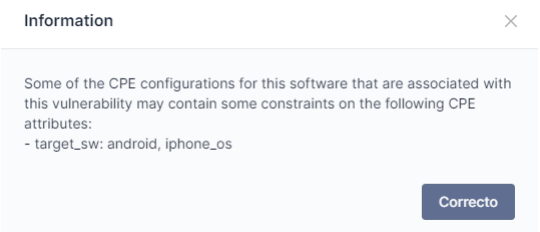

Figura 23: Alerta de vulnerabilidad con restricciones para un programa concreto.

4.6.4. Clases de soporte

A parte de las clases que implementan las diferentes vistas descritas en este apartado y, en caso de haber interacción con la base de datos, las respuestas a las peticiones de los formularios correspondientes, se han implementado dos clases nuevas que proporcionan funciones de apoyo a las anteriores y se han añadido funcionalidades de soporte a algunas de las clases ya mencionadas a lo largo de este capítulo dedicado a la implementación del plugin.

- **cveconnection.class.php**: Este fichero contiene una clase que extiende la antes mencionada clase **PluginNvdConnection**. Esta nueva clase, denominada **PluginNvdCveconnection**, permite realizar conexiones a la API REST de CVE, concretamente al punto de la API "*browse*", que devuelve, en formato json, una lista de los nombres de fabricantes disponibles en el estándar CPE. Si a este mismo punto se le añade el nombre de un fabricante en este estándar, la respuesta devuelve, en el mismo formato, una lista de nombres de productos en el estándar CPE que pertenecen a dicho fabricante.

  Como puede deducirse estas listas se emplearán para poblar los desplegables de fabricantes y productos de la vista de asociaciones programa-nombres CPE definida en el apartado **4.6.2**.

- **cpe.class.php**: A este ya mencionado fichero, que contiene la clase **PluginNvdCpe**, se ha añadido la funcionalidad de, dados los nombres en la base de datos de GLPI de un programa y su fabricante, devolver sendas listas de términos a emplear como recomendaciones para la búsqueda de nombres de los campos "*vendor*" y "*product*" del estándar CPE.

  Esta funcionalidad se empleará, al igual que el caso anterior, en la vista de asociaciones programa-nombres CPE definida en el apartado **4.6.2**. En este caso, sin embargo, las listas obtenidas se utilizarán para poblar los desplegables que muestren sugerencias con las que filtrar los desplegables principales de fabricantes y productos.

- **cverecord.class.php**: En este nuevo fichero se ha creado la clase **PluginNvdCverecord**, que implementa funciones auxiliares para manejar la información de las vulnerabilidades mostradas en las diferentes vistas de vulnerabilidades definidas en el apartado **4.6.3** (General, de dispositivo y de programa). Dichas funcionalidades son:
  - A partir del identificador CVE de una vulnerabilidad, obtener el enlace a su correspondiente registro en la base de datos NVD.
  - A partir de una puntuación CVSS, calcular la severidad de la vulnerabilidad.
  - A partir del idioma en el que se ha configurado la aplicación de GLPI, obtener una descripción de vulnerabilidad almacenada en tal idioma. En caso de no ser posible siempre se muestra en inglés.
  - Generar el mensaje de advertencia para cierta vulnerabilidad en caso de que existan configuraciones con restricciones en los campos CPE.

4.7. Dificultades encontradas

En este último apartado del capítulo dedicado a las fases de investigación y desarrollo del proyecto se detallarán las principales dificultades encontradas durante estas etapas y las aproximaciones que se han seguido para superarlas o bien los motivos por los que no han podido abordarse en el tiempo proyectado para la implementación del proyecto.

4.7.1. Identificación de programas y sistemas operativos

Una de las mayores dificultades encontradas durante el desarrollo del proyecto, como se ha mencionado ya en varios apartados, ha sido encontrar un método para identificar de forma inequívoca las instalaciones de las diferentes versiones de programas y sistemas operativos en los equipos gestionados por la herramienta GLPI. Tras una fase inicial de investigación, se ha llegado a la conclusión de que el método más adecuado se trataba de emplear el estándar CPE, puesto que no sólo permite indicar todos los posibles aspectos que pudieran diferenciar las versiones de un software en concreto, sino que además es el mecanismo que emplean las principales bases de datos de vulnerabilidades públicas para especificar las configuraciones afectadas por las vulnerabilidades que recogen en sus registros.

La principal contrariedad, sin embargo, surge a raíz de las diferencias entre los nombres obtenidos del inventariado software de un equipo y sus correspondientes versiones en el estándar CPE. Esto supone la necesidad de implementar un mecanismo de traducción, a modo diccionario, entre estas dos versiones para poder emplear los inventariados nativos de la herramienta en la búsqueda de vulnerabilidades. Sería necesario pues traducir cada uno de los campos de los nombres CPE (salvo las versiones) para poder utilizar la información del inventariado al completo, sin embargo, por razones de plazos, únicamente se ha implementado la traducción automática de nombres de sistemas operativos, proporcionando un mecanismo de asignación manual para los nombres de fabricantes y productos de programas y dejando de lado información más secundaria como las posibles actualizaciones, ediciones o software y hardware en los que se ejecutan las configuraciones concretas.

Esta aproximación permite efectivamente realizar las búsquedas de vulnerabilidades planteadas en los objetivos del proyecto, informando al usuario cuando las configuraciones asociadas con las vulnerabilidades encontradas contuvieran alguna restricción en los campos del estándar CPE no empleados en las búsquedas (actualización, edición, software y hardware objetivos).

Pese a que esta dificultad, siendo la principal encontrada con respecto a la identificación de programas y sistemas operativos, ha podido resolverse de forma que puedan cumplirse con los objetivos del proyecto, se han encontrado otra serie de dificultades relacionadas con el estándar CPE que no han podido abordarse, bien debido a una falta de recursos o al haberse considerado inviables su resolución por quedar, técnicamente, fuera de los límites en los que actúa este estándar.

- **Uso inconsistente de campos del estándar CPE**: Se ha podido observar, a lo largo del diccionario oficial del estándar CPE [9], casos en los que un mismo producto se identificaba de formas diferentes, cambiando los valores de los campos del estándar CPE. Un ejemplo de este uso inconsistente puede observarse en las versiones con las que se identifican las configuraciones afectadas del producto Microsoft Office en las siguientes vulnerabilidades:
    - CVE-2021-40481 [29]: *cpe:2.3:a:microsoft:office:2019*
    - CVE-2021-43905 [30]: *cpe:2.3:a:microsoft:office:18.2110.13110.0*
    - CVE-2022-21840 [31]: *cpe:2.3:a:microsoft:office:2019*

  Como puede apreciarse, la primera vulnerabilidad y la última emplean el número de la edición del programa (2019) como identificador de la versión, mientras que la segunda emplea el número de versión completo. Esta situación podría llegar a entenderse si se considerara un posible cambio en la nomenclatura del estándar, sin embargo, las vulnerabilidades se han listado en orden de publicación, que coincide a su vez con el orden de las fechas de la última modificación realizada sobre sus correspondientes registros. Así pues, se concluye que existen casos para los cuales no se cumple de manera consistente el estándar CPE para la identificación de las configuraciones afectadas por vulnerabilidades en las bases de datos públicas.

- **Cambios en el estándar CPE**: Una de las posibilidades mencionadas para explicar la inconsistencia en el uso de los campos CPE expuesta en el punto anterior era el posible cambio en la nomenclatura de productos en el estándar CPE. Esta misma posibilidad constituye en sí misma una nueva dificultad para identificar consistentemente los productos para los cuales se desean buscar vulnerabilidades, pues en caso de implementar un mecanismo que tradujera los nombres obtenidos de un inventariado al estándar CPE, habría que actualizarlo con cada cambio de nomenclatura realizado en este estándar.

  Un posible ejemplo de esta dificultad sería los cambios que sufrieron los nombres con los que se identificaban los sistemas operativos Windows. En el caso de Windows 10, concretamente de su versión 22h2, su nombre CPE pasó de ser *cpe:2.3:o:microsoft:windows_10:22h2* a emplear el campo de versión con *cpe:2.3:o:microsoft:windows_10_22h2:**.

  Para el desarrollo del plugin se han tenido en cuenta los cambios de nomenclatura más recientes, no obstante, conforme el estándar se actualice deberán actualizarse los mecanismos de traducción implementados para poder continuar consultando las bases de datos públicas. Otra posible dificultad que aparecerá en este caso será cómo consultar aquellos registros en estas bases de datos cuyas configuraciones afectadas sigan empleando en la nomenclatura no actualizada.

  Ejemplo de registro con la antigua nomenclatura de Windows 10 : CVE-2021-21552 [32].

- **Programas no incluidos en el estándar CPE**: Pese a que el diccionario oficial del estándar CPE (Documento oficial que contiene todos los nombres de programas y fabricantes aprobados para el estándar) es extenso y se encuentra en constante crecimiento, no todos los programas creados se encuentran en él, en especial los menos distribuidos. Por supuesto aquellos programas desarrollados por particulares o ejecutables compilados por los propios usuarios, pese a que aparezcan en los informes de inventariado proporcionados a la herramienta GLPI, no podrán emplearse en búsquedas de vulnerabilidades conocidas, dado que, al no conocerse o no ser públicos, es altamente improbable que la comunidad pueda hallar ninguna vulnerabilidad en su código, sin importar si las haya o no.
- **Versiones de programas no oficiales**: De forma similar al caso anterior, existen versiones de programas oficiales distribuidas de manera no oficial por sitios web de terceros, normalmente, para evitar el pago de las licencias que normalmente se requieren para su uso o, en otros casos, porque han sido adaptadas de manera extraoficial para sistemas para los cuales no habían sido diseñadas (Este es el caso de muchos programas encontrados en distribuciones Linux que originalmente se idearon para Windows).

  En estos casos la información recabada por el agente de inventariado suele no coincidir con aquella disponible en las versiones oficiales de los programas, dificultando o incluso impidiendo el proceso de traducción al estándar CPE para la comprobación de posibles vulnerabilidades en los programas no oficiales.

4.7.2. Registros de vulnerabilidades sin versión especificada

Pese a que la última versión del estándar CPE se publicó en agosto de 2011, sustituyendo a la anterior versión que databa de marzo de 2009, las bases de datos de vulnerabilidades públicas como CVE o NVD todavía almacenan registros de vulnerabilidades publicadas que carecen de configuraciones específicas afectadas por la vulnerabilidad, lo que hace imposible su consulta, o que presentan configuraciones en las que no se especifican las versiones concretas de los programas o sistemas operativos afectados, lo que supone que, al consultar con cualquier versión de un software concreto, siempre se devuelva estas vulnerabilidades, pese a que, más que probablemente, hayan sido resueltas en versiones modernas del programa/sistema operativo.

Un ejemplo de este problema sería la vulnerabilidad CVE-2019-0697 [33], para la cual la configuración concreta del sistema operativo Windows Server 2019 afectado es la siguiente: *cpe:2.3:o:microsoft:windows_server_2019:**. Esta configuración supone que, cualquier posible versión, actualización, edición o arquitectura de implementación de este sistema operativo se considerará vulnerable a una vulnerabilidad que fue publicada en el año 2020, y posiblemente se encuentre parcheada. Para el desarrollo del proyecto se ha decidido incluir estas vulnerabilidades, sin embargo, podría plantearse la posibilidad de únicamente aceptar vulnerabilidades que especificaran en las configuraciones de los registros las versiones software vulnerables.

5. Pruebas y resultados

Para cada una de las funcionalidades diseñadas e implementadas a lo largo del desarrollo de este proyecto, se han realizado pruebas intermedias para delinear la aproximación más adecuada para lograr la implementación final y una vez implementadas con el propósito de comprobar si el producto final cumple con las expectativas y los objetivos marcados en la fase de planificación. Estas pruebas se han llevado a cabo empleando una instalación de XAMPP, que proporciona un servidor web capaz de interpretar PHP y una base de datos MariaDB.

5.1. Consultas a base de datos NVD

Inicialmente, con el fin de comprobar el formato exacto de las respuestas a las consultas a la base de datos NVD, se realizaron una serie de consultas a través del "cmdlet" **Invoke-WebRequest** de PowerShell. La URL puede especificarse mediante el flag **-Uri** y las cabeceras de la petición, que han de contener la clave de la API de NVD, mediante el flag **-Headers**, cuyo valor deberá de ser aportado en formato diccionario (Ejemplo para la clave de la API: @{ 'apiKey' = 'valor' }).

Al analizarse el formato de las diferentes respuestas obtenidas, pudo trasladarse de forma más eficiente la extracción de la información deseada a código PHP y, una vez implementada, depurar su funcionamiento y comprobar que se obtuvieran los resultados esperados.

Una vez se comprobó el funcionamiento de la consulta de información a la base de datos NVD a través del código PHP, se procedió a trasladar este código a una tarea automática, tal y como se ha definido en el apartado 4.3, y comprobar que los resultados obtenidos eran exactamente los mismos que aquellos obtenidos a partir de las consultas manuales. Cabe destacar que la herramienta GLPI proporciona un mecanismo para ejecutar manualmente las tareas automáticas definidas, no siendo necesario programar en el sistema en el que se encuentra la instancia de GLPI un mecanismo de ejecución periódico que active las tareas definidas. Para la depuración de la funcionalidad de la tarea automática se ha empleado la activación manual de la tarea definida, no obstante, una vez finalizada la implementación de la misma, también se ha comprobado su funcionamiento mediante la activación externa a través de el Programador de tareas de Windows, simulando las condiciones de uso de un entorno real.

5.2. Interacción con la base de datos local

Actualmente, en el código de la versión 1.0.0 del plugin (Versión desarrollada para este proyecto), pueden encontrarse más de 60 tipos de consultas diferentes a la base de datos local de GLPI repartidas entre múltiples clases y que, haciendo uso de las tablas nativas y aquellas definidas por el plugin, hacen posible proveer las diferentes funcionalidades consideradas en la fase de planificación.

La programación de estas consultas en el código de las distintas clases PHP que componen el plugin, sin embargo, no se realizó hasta haber podido comprobar su correctitud, es decir, que la sentencia SQL correspondiente a la consulta diseñada era correcta y proporcionaba, sin errores, la información requerida de la base de datos local. Con el objetivo de probar estas consultas antes de ser trasladadas a código PHP, se empleó una interfaz web (**phpmyadmin** [34]) con la que interactuar con la instancia local de la base de datos, concretamente con el esquema empleado por la herramienta GLPI, pudiendo así ejecutar de forma manual cualquier consulta que fuera a implementarse posteriormente en el plugin desarrollado y comprobar su validez.

Para poder comprobar correctamente la validez de las consultas diseñadas, fue necesario poblar temporalmente de registros de prueba las diferentes tablas a emplear de la base de datos, tanto aquellas nativas al framework de GLPI como aquellas creadas por el plugin. De no introducir esta serie de datos de prueba, únicamente se podría haber validado la sintaxis de las consultas, y no la correctitud de la información obtenida en sí.

Por último, una vez pudo corroborarse el correcto funcionamiento de las consultas manuales, se trasladaron a PHP para poderse ejecutar desde el plugin. La ejecución de cada una de estas consultas en el plugin permitió constatar la correcta implementación de las mismas.

5.3. Procesado del inventariado de GLPI

Hasta el momento, el inventario de activos (Equipos, sistemas operativos y programas) empleado para las pruebas realizadas consistía únicamente de instancias de prueba creadas manualmente para validar una funcionalidad en concreto. No obstante, una de las pruebas más relevantes realizadas durante el proceso de desarrollo ha sido la de procesar inventariados procedentes de equipos informáticos reales, cuando ha sido posible, o máquinas virtuales en su defecto. Estos informes de inventariado, generados mediante el agente de inventariado de GLPI, proporcionan información equivalente a la que podría esperarse obtener en un entorno real de uso de la herramienta.

La muestra empleada para las pruebas se compone de 9 equipos con instalaciones de sistemas operativos de diferentes familias y un repertorio de aplicaciones variado:

- Equipo Windows 10 con aplicaciones orientadas a la informática.
- Equipo Windows 10 con programas de videojuegos.
- Equipo Windows 11 con aplicaciones de ofimática estándar.
- Equipo Windows Server 2019 con instalaciones básicas de AD e IIS.
- Equipo Kali Linux. Instalación por defecto.
- Equipo Debian. Instalación por defecto.
- Equipo Ubuntu. Instalación por defecto.
- Equipo MacOS con aplicaciones orientadas a la informática.
- Equipo MacOS con aplicaciones de ofimática estándar.

Como puede observarse, el objetivo de la muestra escogida era obtener informes de inventariado de equipos con instalaciones de los principales sistemas operativos disponibles en el mercado (Windows y MacOS) y diferentes distribuciones de la familia Linux. Debido a lo gravoso que resulta el análisis manual de un informe de inventariado, no se ha ampliado la muestra con otros sistemas operativos, extrapolando en muchos casos la información obtenida a otras ediciones de sistemas operativos de la misma familia (Otras versiones de Windows / Windows Server o distribuciones Linux similares), sin embargo, podría ser interesante analizar muestras de equipos con sistemas operativos como Ubuntu Server, Fedora o CentOS entre otros.

Tras importar los informes de inventariado generados a GLPI, se han podido probar las funcionalidades de interacción con la base de datos local y consulta automática de vulnerabilidades al base de datos NVD, con los registros creados en esta, empleando información de aplicaciones y sistemas operativos reales, tal y como sucedería en un caso real de uso del plugin. Concretamente se ha empleado para estas pruebas la información de los sistemas operativos de los 9 equipos de la muestra y 15 aplicaciones seleccionadas para cada uno de estos equipos.

Para las pruebas se ha tratado de emplear la información sobre las versiones de los sistemas operativos y aplicaciones contenida en las tablas nativas de GLPI y, a través de un mínimo procesado general, conseguir componer exitosamente los correspondientes nombres CPE y realizar consultas a la base de datos NVD. Los resultados de estas pruebas se pueden observar en la **Figura 24** y la **Figura 25**. Estas figuras muestran como tan solo 2 de los 9 (22%) sistemas operativos (los correspondientes a los equipos MacOS) han podido ser identificados correctamente, mientras que, de las 135 aplicaciones presentes en la muestra, únicamente se han obtenido los nombres CPE correctos de 24 (18%).

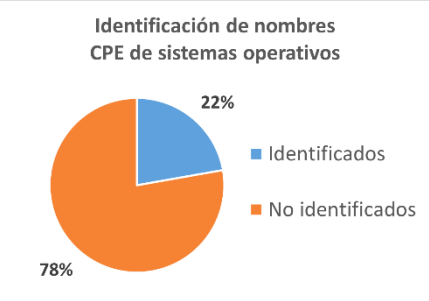

Figura 24: Estadísticas de identificación de sistemas operativos empleando los nombres presentes en los informes de inventariado.

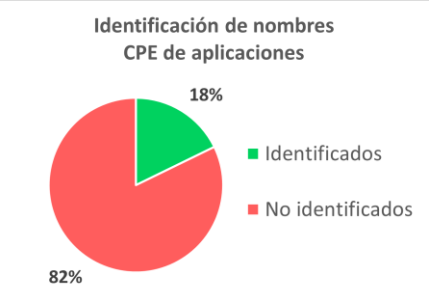

Figura 25: Estadísticas de identificación de aplicaciones empleando los nombres presentes en los informes de inventariado.

Estas pruebas han permitido determinar la necesidad, como se ha mencionado ya en repetidas ocasiones en el capítulo **4**, de implementar un mecanismo que permita traducir los nombres de productos y fabricantes de las aplicaciones y sistemas operativos contenidos en los informes de inventariado a sus correspondientes versiones en el estándar CPE, con el fin de poder emplear esta información para buscar vulnerabilidades en los equipos de los cuales se han obtenido dichos informes.

5.4. Vistas

Los tres tipos de vistas diferentes añadidas por el plugin a la herramienta GLPI fueron diseñadas de forma previa empleando etiquetas HTML, estilos CSS y funciones de JavaScript contenidos de forma completa en ficheros PHP y siendo depuradas en el mismo servidor web en el que se realizó la instalación de la instancia de GLPI. Una vez se consiguió la estructura deseada se procedió a trasladar estas vistas a las diferentes clases definidas para su implementación en el plugin, separando la estructura, estilos y funciones en sus correspondientes archivos (clases PHP, hoja de estilos CSS y archivo de funciones JS).

5.5. Pruebas finales

Una vez finalizada la implementación de las distintas funcionalidades del plugin tras las pruebas intermedias realizadas, se procedió a probar en su conjunto la interacción con el plugin a través de las vistas creadas y la capacidad para identificar, a través de los mecanismos de traducción y asociación de nombres implementados, los nombres CPE de los mismos 9 sistemas operativos y 135 aplicaciones empleados en las pruebas de inventariado del apartado **5.3**. Los resultados de estas pruebas se muestran en la **Figura 26** y la **Figura 27**.

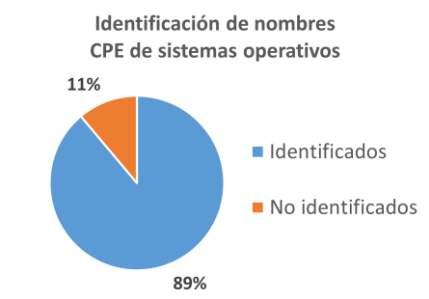

Figura 26: Estadísticas de identificación de sistemas operativos empleando el mecanismo de traducción implementado.

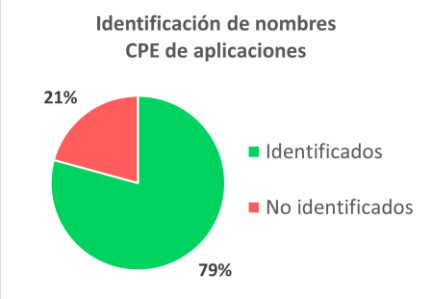

Figura 27: Estadísticas de identificación de aplicaciones empleando los nombres CPE de las asociaciones creadas.

Como puede observarse, los mecanismos de traducción y asociación de nombres CPE implementados han permitido mejorar considerablemente los resultados con respecto a las pruebas realizadas únicamente con los nombres disponibles en los reportes de inventariado. Únicamente uno de los 9 sistemas operativos empleados no ha podido ser identificado por el sistema de traducción implementado, mientras que tan sólo 28 de las 135 aplicaciones han quedado fuera del conjunto de las aplicaciones correctamente traducidas al estándar CPE. Los casos en los que no se ha podido identificar el nombre CPE de las aplicaciones/sistema operativo se deben a que estos no se encuentran añadidos al estándar en el momento en el que se ha entregado este proyecto (junio 2023).

Cabe destacar que, de las aplicaciones analizadas en los diferentes equipos de la muestra, únicamente aquellas recogidas en los inventariados de sistemas Windows contenían información completa y correcta de los fabricantes de las mismas, faltando en muchas ocasiones esta información en las aplicaciones obtenidas de equipos MacOS o siendo errónea en los equipos de distribuciones Linux, en los que se define como el nombre de la distribución.

De igual manera se ha de resaltar que, para buena parte de las aplicaciones instaladas en los equipos de distribuciones Linux, las versiones recogidas por el agente de inventariado requieren de un procesamiento para poder cumplir con aquellas incluidas en el estándar CPE. Esto se debe a que muchas de estas aplicaciones provienen de repositorios que añaden información a los números de versión que los hace incompatibles con el estándar CPE. Un ejemplo podría ser la versión 3.0 de **cron** en Ubuntu, que se muestra como "*3.0pl1-137ubuntu3*" cuando su nombre CPE debería ser "*cron:3.0:pl1-137*".

Tras realizar manualmente la asociación de los nombres CPE correspondientes a las aplicaciones de la muestra seleccionada, se ha ejecutado manualmente la tarea de búsqueda de vulnerabilidades para comprobar el funcionamiento conjunto de las funcionalidades implementadas. Los resultados, representados en la **Figura 28**, muestran un total de 2673 vulnerabilidades distintas encontradas para las distintas instalaciones de las aplicaciones y sistemas operativos de la muestra. De estas vulnerabilidades 927 se corresponden con alguna configuración de programas, 1739 con alguna versión concreta de un sistema operativo, y 7 afectan tanto a alguna aplicación como a algún sistema operativo. Esta distribución, así como la información relacionada con las vulnerabilidades encontradas para la muestra, puede apreciarse a través de la vista general de vulnerabilidades desarrollada para el plugin, tal y como se muestra en la **Figura 29**.

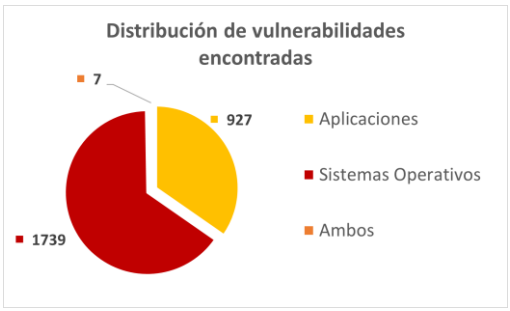

Figura 28: Distribución de las vulnerabilidades encontradas según el tipo de configuración a la que afectan.

Figura 29: Información de las vulnerabilidades encontrades para la muestra.

6. Conclusiones y trabajos futuros

En este capítulo se exponen las conclusiones extraídas del desarrollo del proyecto y las pruebas de funcionalidad realizadas durante el mismo y tras su finalización. Tras esto se proponen posibles líneas de trabajo con las que mejorar las funcionalidades ya implementadas y añadir nuevas funcionalidades al plugin en un futuro.

6.1. Conclusiones principales

En este proyecto se ha desarrollado un plugin, cuyo código se encuentra disponible de forma gratuita bajo la licencia GNU GPLv3, para la herramienta de gestión de activos informáticos GLPI. El plugin en cuestión permite, haciendo uso del inventariado de equipos gestionados por GLPI, encontrar aquellas vulnerabilidades publicadas que afectan a los sistemas operativos y programas instalados en estos. Para ello el plugin hace uso de los recursos proporcionados por el framework de GLPI ofreciendo al usuario de la herramienta una interfaz con la que configurar el comportamiento del mismo, establecer las asociaciones necesarias para identificar correctamente las aplicaciones de las cuales se desea buscar vulnerabilidades y visualizar de manera intuitiva la información correspondiente a las vulnerabilidades encontradas. Para la identificación de sistemas operativos se ha implementado un mecanismo automático que reconoce mediante patrones las diferentes familias, distribuciones y versiones de los principales sistemas operativos del mercado, lo que permite obtener en la gran mayoría de los casos una identificación exitosa. Por el contrario, los resultados de las pruebas realizadas durante la implementación del plugin han mostrado la gran dificultad que existe a la hora de identificar los nombres de las aplicaciones, sus fabricantes y sus versiones en muchos casos (especialmente en sistemas Linux o MacOS). Este hecho ha supuesto la necesidad de implementar un mecanismo de asociación manual entre aplicaciones y sus correspondientes nombres en el estándar empleado para la identificación inequívoca de las instalaciones, lo que, por un lado, supone que el plugin, en teoría, es capaz de identificar cualquier aplicación que se encuentre incluida en el estándar una vez se haya realizado la correspondiente asignación de nombres, pero, por otro lado, implica una relativamente gravosa tarea para el usuario encargado de realizar estas asignaciones. El funcionamiento final del plugin se ha comprobado empleando inventariados de una muestra de equipos Windows, MacOS y Linux, constatándose la capacidad para encontrar vulnerabilidades y mostrar la información relativa a estas, a la vez que confirmando la necesidad de, en un futuro, pulir los mecanismos de identificación de instalaciones para mejorar la precisión de las búsquedas realizadas.

En lo relativo a los objetivos planteados en la fase de planificación del proyecto, se ha de concluir que se han cumplido todos y cada uno de ellos en su totalidad a excepción de uno, el referente a la clasificación de las vulnerabilidades encontradas. En un principio se planteó clasificar las vulnerabilidades en función de su criticidad y tipo de vulnerabilidad y equipo en el que se hubieran encontrado, sin embargo, una vez finalizada la implementación del plugin, únicamente se clasifican atendiendo a la criticidad.

La principal razón por la que no ha podido implementarse la clasificación de vulnerabilidades en función de su tipo y la clase de equipo a la que afectaran ha sido por los plazos del proyecto. A lo largo de la fase de diseño e implementación se han encontrado una serie de dificultades que obligaron a modificar parte de la planificación original, resultando en más tiempo invertido en funcionalidades base del plugin y menos tiempo restante para poder implementar las funcionalidades finales. Aún con todo, gracias a la metodología seguida, se ha logrado alcanzar la gran mayoría de los objetivos planteados al inicio del proyecto y entregar en plazo el mismo, pudiendo plantear esta funcionalidad pendiente como una de las vías de trabajo futuro a desarrollar con el plugin.

6.2. Trabajo a futuro

A continuación, se plantean una serie de ideas sobre las que poder trabajar en un futuro con el objetivo de expandir las funcionalidades del plugin desarrollado.

- **Traducción automática al estándar CPE**: La mayor dificultad encontrada durante el desarrollo del plugin y, por lo tanto, la principal prioridad para expandir la funcionalidad del plugin. Esta vía de desarrollo implicaría implementar un mecanismo automático que reconociera la información de las aplicaciones según se almacena por GLPI y la tradujera automáticamente al estándar CPE sin la necesidad de darse una interacción previa con un usuario. El gran reto pasa por reconocer las instalaciones en sistemas Linux y MacOS, en los que se ha podido comprobar la falta o incorrección de la información almacenada sobre las aplicaciones, no obstante, el caso de los sistemas Windows parece más prometedor en este sentido.

  De forma similar se habría de implementar la traducción para aquellos campos del estándar CPE que no se han empleado en las consultas (*update*, *edition*, *target software* y *target hardware*), lo que permitiría mejorar la calidad de las mismas y la precisión de la información recabada.

  Por último, se habría de añadir más distribuciones al ya implementado mecanismo de identificación de sistemas operativos, como sistema operativos para móviles (Android e IOS), otros de la rama Linux (CentOS, ArchLinux, Parrot, Kali, Manjaro) o algunos de la rama de Unix (Solaris, OpenBSD, FreeBSD).

- **Vulnerabilidades del kernel del sistema operativo**: Si bien actualmente el plugin es capaz de identificar versiones de sistemas operativos y buscar vulnerabilidades relacionadas con estas, no se emplea la información referente al kernel de las distribuciones MacOS o Linux, que puede presentar una serie de vulnerabilidades asociadas a las versiones del mismo. Podría plantearse la posibilidad de investigar acerca de las vulnerabilidades que afectan a estas versiones para comprobar si siempre van ligadas a las implementaciones de las distintas distribuciones que se construyen sobre estos núcleos, en cuyo caso no aportarían nueva información, o por el contrario existen vulnerabilidades publicadas independientemente de las distribuciones.

- **Mejorar las vistas implementadas**: Si bien las vistas implementadas para mostrar información sobre las vulnerabilidades encontradas cumplen con los objetivos planteados para el proyecto, podrían mejorarse añadiendo información y funcionalidades. Un tipo de información relevante a añadir a las tablas de vulnerabilidades sería el correspondiente identificador CWE que aportara información sobre la clase de vulnerabilidad mostrada. También podría ser interesante que la vista de vulnerabilidades de un programa en concreto mostrara una columna que contuviera, para cada vulnerabilidad, una lista de los equipos en los que se encuentra presente.

  Por último, la principal mejora planteada para las vistas creadas podría ser permitir a los usuarios filtrar las vulnerabilidades mostradas según el contenido de cada una de las columnas de la tabla. Esto facilitaría enormemente la visualización de las vulnerabilidades y la gestión de las mismas.

- **Búsquedas de vulnerabilidades reducidas**: Actualmente la búsqueda de nuevas vulnerabilidades y la actualización o eliminación de aquellas ya presentes corren a cargo de la tarea automática programada con este fin, cuya periodicidad de ejecución puede ser definida por los administradores de la herramienta GLPI. Durante las últimas pruebas realizadas se comprobó que esta tarea, para una muestra lo suficientemente grande, puede demorarse decenas de minutos incluso unas pocas horas hasta completarse. Podría darse la necesidad puntual de, tras una nueva instalación o actualización de una aplicación o sistema operativo concreto, buscar vulnerabilidades relacionadas con la nueva versión, sin embargo, ejecutar manualmente la tarea de búsqueda de vulnerabilidades podría resultar muy costoso cuando únicamente se requiere comprobar un número reducido de instalaciones. Es por esto por lo que se plantea la posibilidad de añadir una nueva funcionalidad que permita ejecutar el algoritmo de búsqueda y actualización de vulnerabilidades para únicamente una versión o conjunto de versiones reducido. Las ejecuciones podrían lanzarse desde los correspondientes puntos de la interfaz de GLPI, es decir, si se deseara, por ejemplo, buscar vulnerabilidades relacionadas con un programa en concreto, se habría de acceder a la página del programa para ejecutar el escaneo, mientras que, si se quisiese hacer lo mismo para un sistema operativo, habría que lanzar la búsqueda desde la página de equipo en el que estuviera instalado.

- **Mantenimiento de las vulnerabilidades almacenadas**: Los registros de vulnerabilidades de las bases de datos públicas pueden sufrir modificaciones a lo largo del tiempo, por lo que sería interesante implementar una tarea que, periódicamente, recabara información sobre aquellas vulnerabilidades almacenadas que han sufrido una actualización desde la última comprobación, modificando la información pertinente en la base de datos local en caso de ser necesario. De forma similar una vulnerabilidad puede ser rechazada en cierto punto, por lo que también se deberían de eliminar de la base de datos aquellas vulnerabilidades que hubieran sido rechazadas.

7. Glosario

- GLPI: Gestionnaire Libre de Parc Informatique (Gestor libre de equipos informáticos).
- CVE: Common Vulneravilities and Exposures (Vulnerabilidades y exposiciones comunes).
- MITRE: Massachusetts Institute of Technology Research and Engineering (Instituto de investigación tecnológica e ingeniería de Massachusetts).
- CWE: Common Weakness Enumeration (Listado de debilidades comunes).
- CPE: Common Platform Enumeration (Listado de plataformas comunes).
- NIST: National Institute of Standards and Technology (Instituto nacional de estándares y tecnología).
- NVD: National Vulnerability Database (Base de datos nacional de vulnerabilidades).

8. Bibliografía

[1] [En línea]. Available: https://glpi-project.org/. [Último acceso: 13 Marzo 2022].

[2] TECLIB. [En línea]. Available: https://plugins.glpi-project.org/#/plugin/openvas. [Último acceso: 13 Marzo 2022].

[3] L. B. Sanguino. [En línea]. Available: https://github.com/lbenthins/iva. [Último acceso: 13 Marzo 2022].

[4] C. Conard. [En línea]. Available: https://github.com/cconard96/glpi-cve-plugin. [Último acceso: 13 Marzo 2022].

[5] [En línea]. Available: https://www.un.org/sustainabledevelopment/es/. [Último acceso: 13 Marzo 2022].

[6] TECLIB, «Plugins-GLPI,» [En línea]. Available: https://plugins.glpi-project.org/#/. [Último acceso: 11 Abril 2023].

[7] MITRE, «CVE MITRE,» [En línea]. Available: https://cve.mitre.org/. [Último acceso: 11 Abril 2023].

[8] CIRCL, «CVE Api,» [En línea]. Available: https://cve.circl.lu/api/. [Último acceso: 9 Mayo 2023].

[9] NIST, «CPE NIST,» [En línea]. Available: https://nvd.nist.gov/products/cpe. [Último acceso: 11 Abril 2023].

[10] NIST, «NVD NIST,» [En línea]. Available: https://nvd.nist.gov/. [Último acceso: 11 Abril 2023].

[11] I. FIRST.ORG, «CVSS,» [En línea]. Available: https://www.first.org/cvss/. [Último acceso: 9 Mayo 2023].

[12] MITRE, «CWE,» [En línea]. Available: https://cwe.mitre.org/. [Último acceso: 9 Mayo 2023].

[13] NIST, «NVD API,» [En línea]. Available: https://nvd.nist.gov/developers/vulnerabilities. [Último acceso: 9 Mayo 2023].

[14] TECLIB, «GLPI Doc Install,» [En línea]. Available: https://glpi-install.readthedocs.io/en/latest/index.html. [Último acceso: 11 Abril 2023].

[15] TECLIB, «GLPI Doc Dev,» [En línea]. Available: https://glpi-developer-documentation.readthedocs.io/en/master/plugins/index.html. [Último acceso: 11 Abril 2023].

[16] TECLIB, «GLPI Doc Plugins,» [En línea]. Available: https://glpi-install.readthedocs.io/en/latest/index.html. [Último acceso: 11 Abril 2023].

[17] «XAMPP,» [En línea]. Available: https://www.apachefriends.org/. [Último acceso: 11 Abril 2023].

[18] C. B. González, «GitHub,» [En línea]. Available: https://github.com/Carlos-Borau/GLPI-NVD-PLUGIN. [Último acceso: 11 Abril 2023].

[19] TECLIB, «Generador de consultas BD GLPI,» [En línea]. Available: https://glpi-developer-documentation.readthedocs.io/en/master/devapi/database/dbiterator.html. [Último acceso: 9 Mayo 2023].

[20] PHP, «cURL PHP,» [En línea]. Available: https://www.php.net/manual/en/book.curl.php. [Último acceso: 9 Mayo 2023].

[21] TECLIB, «GLPI Automatic Actions,» [En línea]. Available: https://glpi-developer-documentation.readthedocs.io/en/master/devapi/crontasks.html. [Último acceso: 9 Mayo 2023].

[22] TECLIB, «GLPI Plugin Database,» [En línea]. Available: https://glpi-developer-documentation.readthedocs.io/en/master/plugins/database.html. [Último acceso: 9 Mayo 2023].

[23] «dbdiagram,» [En línea]. Available: https://dbdiagram.io/home. [Último acceso: 6 Junio 2023].

[24] TECLIB, «GLPI Inventory Agent,» [En línea]. Available: https://github.com/glpi-project/glpi-agent/releases/tag/1.4. [Último acceso: 9 Mayo 2023].

[25] TECLIB, «GLPI Agent Docs,» [En línea]. Available: https://glpi-agent.readthedocs.io/en/latest/. [Último acceso: 9 Mayo 2023].

[26] Teclib, «GLPI agent nightly builds,» [En línea]. Available: https://nightly.glpi-project.org/glpi-agent/. [Último acceso: 6 Junio 2023].

[27] «GLPI agent nightly build 1.5,» [En línea]. Available: https://nightly.glpi-project.org/glpi-agent/#1-5-git73ef762f. [Último acceso: 6 Junio 2023].

[28] NIST, «NIST NVD API key request,» [En línea]. Available: https://nvd.nist.gov/developers/request-an-api-key. [Último acceso: 6 Junio 2023].

[29] NIST, «NVD NIST CVE-2021-40481,» [En línea]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-40481. [Último acceso: 9 Junio 2023].

[30] NIST, «NVD NIST CVE-2021-43905,» [En línea]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-43905. [Último acceso: 9 Junio 2023].

[31] NIST, «NVD NIST CVE-2022-21840,» [En línea]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-21840. [Último acceso: 9 Junio 2023].

[32] NIST, «NVD NIST CVE-2021-21552,» [En línea]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-21552. [Último acceso: 9 Junio 2023].

[33] NIST, «NVD NIST CVE-2019-0697,» [En línea]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-0697. [Último acceso: 9 Junio 2023].

[34] phpmyadmin, «PhpMyAdmin,» [En línea]. Available: https://www.phpmyadmin.net/. [Último acceso: 11 Junio 2023].

9. Anexos

9.1. Instalación y configuración del plugin

El proceso de instalación y configuración del plugin desarrollado se encuentra detallado en el documento **README.md** del repositorio oficial del proyecto [18], no obstante, se ha decidido incluir los pasos al detalle en este anexo.

Instalación

1. Descargar el contenido del repositorio en forma de archivo comprimido o a través del cliente de GitHub.
2. En caso de haber descargado el código en formato comprimido descomprimir los contenidos del archivo zip.
3. Una vez se tiene el contenido del repositorio en un directorio local, renombrar dicho directorio con el nombre "**nvd**".
4. Mover el directorio "**nvd**" y la totalidad de su contenido a la carpeta "*plugins*" localizada en el directorio raíz de GLPI.
5. Acceder a la instancia de GLPI a través de la interfaz web y navegar al menú de configuración de los plugins (Configuración > Plugins).
6. Localizar el plugin nvd e instalarlo haciendo click sobre el icono de carpeta con símbolo "**+**".
7. Una vez instalado activar el plugin con el interruptor al lado del icono, antes de instalación, ahora de desinstalación.
8. Al tornarse verde el interruptor el plugin se encontrará activado.

Configuración

9. La clave de la API de NVD puede configurarse desde el menú Configuración > General > NVD Plugin.
10. Para realizar la asociación entre aplicaciones y sus correspondientes nombres en el estándar CPE habrá que acceder a la vista de la aplicación en cuestión (Activos > Software > PROGRAMA > CPE Association).
11. En la vista de asociación se muestran 2 desplegables inferiores en los que escoger los nombres de fabricante y producto a asociar a un programa. Estas listas pueden filtrarse empleando filtros manuales o las sugerencias mostradas en los desplegables superiores.
12. Una vez seleccionados los nombres deseados para actualizarlos únicamente se ha de pulsar el botón de confirmación inferior.
13. La periodicidad de la tarea de búsqueda de vulnerabilidades puede configurarse desde el menú de tareas automáticas (Configuración > Acciones automáticas). La tarea tiene el nombre **UpdateVulnTask**. Desde este mismo menú también puede lanzarse la tarea de forma manual.