VNiVERSiDAD D SALAMANCA

TRABAJO DE FIN DE MÁSTER

MÁSTER EN INGENIERÍA INFORMÁTICA

FACULTAD DE CIENCIAS

ANEXO 4: Plan de seguridad

Sistema de traducción asistida con preservación de composición documental

Computer-assisted translation system with preservation of document layout

Septiembre 2025

Autor: Pablo Caño Pascual

Tutores:

Pablo Chamoso Santos

Guillermo Hernández González

Contenido

1. Introducción

El presente anexo detalla el Plan de Seguridad concebido para el sistema de traducción asistida. Su objetivo primordial es establecer las estrategias y medidas adoptadas para garantizar la **confidencialidad, integridad y disponibilidad** tanto de la aplicación como, fundamentalmente, de los datos procesados: los documentos subidos por los usuarios, que pueden contener información sensible. La seguridad, por tanto, se considera un requisito no funcional transversal y crítico para la fiabilidad y aceptación del sistema.

En lugar de considerar la seguridad como una capa añadida al final del desarrollo, este proyecto ha adoptado un enfoque de **Seguridad por Diseño** (*Security by Design*). Esto implica que las consideraciones de seguridad han influido en las decisiones arquitectónicas y tecnológicas desde las primeras fases. La elección de una arquitectura de microservicios contenerizada, descrita en el Anexo 2, no solo responde a requisitos de escalabilidad y modularidad, sino que también establece un principio de aislamiento de componentes que limita el impacto de posibles brechas de seguridad. De igual manera, patrones como el procesamiento asíncrono y el almacenamiento desacoplado contribuyen a mitigar riesgos y a proteger los datos de forma inherente a la estructura del sistema.

Este documento se estructura en varias secciones para abordar de manera integral los diferentes aspectos de la seguridad del sistema:

- **Metodologías y estrategias:** Se describen los enfoques generales empleados para la construcción de un software seguro, incluyendo la validación de entradas y la gestión de dependencias.
- **Protección de datos:** Se detalla cómo se protege la información del usuario tanto en tránsito como en reposo, con especial énfasis en el mecanismo de control de acceso a los ficheros.
- **Identificación de componentes críticos:** Se analizan los módulos del sistema más sensibles desde una perspectiva de seguridad y las medidas específicas implementadas en cada uno.

- **Marco legal y normativo:** Se exponen las consideraciones legales relevantes, principalmente en lo que respecta al cumplimiento del Reglamento General de Protección de Datos (RGPD).

2. Metodologías y estrategias para un software seguro

Para garantizar la robustez del sistema frente a posibles amenazas, se han integrado diversas estrategias de seguridad a lo largo de todo el ciclo de vida del proyecto. Este enfoque proactivo se centra en la prevención y mitigación de vulnerabilidades desde el diseño hasta el despliegue.

Aunque no se ha realizado un análisis de amenazas exhaustivo bajo un marco formal como STRIDE, el diseño del sistema se ha guiado por un modelo de amenazas simplificado. Este proceso consistió en identificar los activos críticos y los principales vectores de ataque que podrían comprometerlos.

- **Activos clave a proteger:**
    - **Documentos del usuario:** Tanto los archivos PDF originales como los traducidos, que constituyen el activo más sensible.
    - **Datos de traducción:** Los ficheros JSON intermedios que contienen el texto extraído y traducido.
    - **Credenciales de servicios externos:** Principalmente, la clave de API utilizada para la comunicación con OpenRouter.
- **Vectores de ataque considerados:**
    - **Acceso no autorizado a datos:** Intentos por parte de un actor malicioso de acceder a documentos que no le pertenecen.
    - **Inyección de datos maliciosos:** Carga de archivos corruptos o manipulación de las peticiones a la API para intentar explotar vulnerabilidades en el backend o en los workers.
    - **Denegación de servicio (DoS):** Intentos de sobrecargar el sistema mediante la subida masiva o concurrente de documentos de gran tamaño para agotar los recursos computacionales.

2.1 Seguridad en el ciclo de vida del desarrollo

Las buenas prácticas de seguridad se han incorporado directamente en el proceso de desarrollo del software:

- **Validación de entradas:** se aplica una validación estricta en todos los puntos de entrada de datos para mitigar los riesgos de inyección y manipulación.
    - **En el Frontend:** la librería **Zod** se utiliza para validar los datos del formulario (selección de idioma, modelo, etc.) antes de que la petición sea enviada al servidor. Esto proporciona una primera capa de defensa y mejora la experiencia de usuario al ofrecer retroalimentación inmediata.
    - **En el Backend API: Pydantic**, integrado nativamente en FastAPI, se encarga de validar automáticamente la estructura y el tipo de todos los datos entrantes en las peticiones HTTP. Cualquier solicitud que no se ajuste al esquema definido es rechazada de forma inmediata, previniendo una amplia gama de ataques basados en la manipulación de entradas.
- **Gestión de dependencias:** el sistema se construye sobre un ecosistema de librerías de código abierto (FastAPI, Vue.js, Celery, etc.). Se reconoce que estas dependencias pueden contener vulnerabilidades conocidas (CVEs). La estrategia de mitigación se basa en la práctica de mantener las dependencias actualizadas a sus últimas versiones estables, aplicando los parches de seguridad que los mantenedores publican.
- **Aislamiento de servicios:** el uso de **Docker** y Docker Compose es una medida de seguridad fundamental. Cada componente del sistema (frontend, backend, worker, etc.) se ejecuta en su propio contenedor aislado. Este aislamiento actúa como un mecanismo de contención: si un componente se viera comprometido (por ejemplo, un worker al procesar un PDF malicioso), el impacto quedaría, en principio, confinado a ese

contenedor, dificultando enormemente que un atacante pueda moverse lateralmente hacia otros servicios o al sistema operativo anfitrión.

2.2 Seguridad en la infraestructura de despliegue

La configuración de la infraestructura de producción ha sido diseñada para minimizar la superficie de ataque y proteger las comunicaciones:

- **Configuración de red segura:** el servidor web **Nginx**, configurado como *reverse proxy*, actúa como el único punto de entrada a la aplicación. Los servicios internos como el backend de FastAPI, MinIO o Redis no están expuestos directamente a Internet; residen en una red privada de Docker. Nginx se encarga de recibir, filtrar y enrutar el tráfico legítimo al servicio correspondiente, protegiendo la infraestructura interna.
- **Mecanismos de comunicación cifrada:** toda la comunicación entre el navegador del usuario y el servidor se realiza de forma cifrada, tal y como se detalla en la siguiente sección.

3. Protección de los datos del usuario

La protección de los documentos de los usuarios es la máxima prioridad del plan de seguridad. Se han implementado medidas específicas para salvaguardar los datos en cada una de sus fases: mientras se transfieren a través de la red (en tránsito) y una vez que están almacenados en la infraestructura del servidor (en reposo).

3.1 Datos en tránsito

Toda la comunicación entre el navegador del usuario y el sistema está protegida mediante cifrado de extremo a extremo para prevenir la interceptación y manipulación de datos.

- **Cifrado de extremo a extremo con HTTPS:** El sistema fuerza el uso del protocolo HTTPS (Hypertext Transfer Protocol Secure) para todas las conexiones. Esto se logra implementando TLS/SSL (Transport Layer Security/Secure Sockets Layer), que cifra la totalidad del tráfico, incluyendo la subida del documento PDF, los parámetros de la traducción y la descarga del resultado. Cualquier intento de conexión a través del puerto 80 (HTTP) es redirigido automáticamente al puerto 443 (HTTPS).
- **Gestión de Certificados:** La gestión de los certificados SSL/TLS se automatiza mediante el uso de **Certbot** y la autoridad de certificación **Let's Encrypt**. El servidor web Nginx está configurado para gestionar el proceso de validación y renovación de certificados de forma automática, garantizando que el cifrado se mantenga siempre activo y actualizado sin intervención manual.

3.2 Datos en reposo

Una vez que los documentos y los datos de traducción asociados llegan al servidor, se almacenan de forma segura utilizando una estrategia de almacenamiento de objetos controlada.

- **Almacenamiento en MinIO:** Todos los ficheros (PDFs originales y traducidos, metadatos JSON) se almacenan en un *bucket* del servicio de almacenamiento de objetos MinIO. De forma crucial, este *bucket* no es de acceso público. El acceso directo a los ficheros desde Internet está denegado por defecto.
- **Control de Acceso mediante URLs Prefirmadas (*Pre-signed URLs*):** Para permitir que el usuario acceda a sus documentos de forma segura, el sistema implementa un patrón de URLs prefirmadas. El flujo es el siguiente:
    1. El frontend solicita al backend API el acceso a un fichero específico (ej. el PDF traducido) a través de un endpoint protegido.
    2. El backend valida la solicitud y se comunica internamente con MinIO para generar una URL de acceso única y temporal para ese fichero.
    3. Esta URL contiene credenciales de acceso firmadas digitalmente y tiene un tiempo de expiración muy corto (p. ej., una hora). Una vez expirada, la URL se vuelve inválida.
    4. El backend responde al frontend con esta URL temporal, a la que el navegador del usuario es redirigido para descargar o visualizar el documento directamente desde MinIO.
       Este mecanismo garantiza que el acceso a los datos sea siempre efímero, autorizado y controlado por la lógica de la aplicación, mitigando eficazmente el riesgo de acceso no autorizado.
- **Aislamiento de datos por tarea:** Como se describe en el Anexo 2, todos los artefactos de un proceso de traducción se almacenan en una estructura de directorios que utiliza el task_id único como prefijo. Esto crea una

segmentación lógica que aísla los datos de cada tarea, impidiendo que un proceso pueda acceder accidentalmente a los ficheros de otro.

3.3 Gestión de secretos y credenciales

La seguridad del sistema también depende de la correcta gestión de las credenciales necesarias para su funcionamiento, como la clave de la API de OpenRouter.

- **Uso de variables de entorno:** Para evitar la exposición de información sensible, las credenciales nunca se escriben directamente en el código fuente (*hardcoding*). En su lugar, se gestionan como **variables de entorno**. El archivo docker-compose.yml se encarga de inyectar estas variables en los contenedores correspondientes (en este caso, el worker de Celery) durante el arranque. Este enfoque permite que el código sea independiente de las credenciales y que estas puedan ser gestionadas de forma segura en el entorno de producción, fuera del control de versiones.

4. Identificación de componentes críticos y medidas específicas

Si bien la seguridad se ha aplicado de manera holística en todo el sistema, ciertos componentes, por su función o por los datos que manejan, representan puntos de mayor criticidad. Un fallo de seguridad en estas áreas podría tener un impacto significativamente mayor. A continuación, se identifican estos componentes y se describen las medidas de seguridad específicas implementadas para protegerlos.

Backend API (FastAPI)

- **Criticidad: Alta.** El backend API es el punto de entrada principal del sistema y el guardián de la lógica de negocio. Es el componente que recibe directamente las peticiones de los usuarios, gestiona la orquestación de tareas y autoriza el acceso a los datos. Una vulnerabilidad en este servicio podría comprometer todo el flujo de trabajo y exponer los endpoints de gestión de tareas.
- **Medidas específicas:**
    - **Validación estricta de esquemas:** Se utiliza Pydantic para validar rigurosamente todas las entradas de la API. Cualquier petición que no se ajuste al esquema esperado es rechazada de inmediato, mitigando riesgos de inyección de datos (ej. NoSQL injection) y otros ataques basados en entradas malformadas.
    - **Limitación del tamaño de archivos:** El servidor web y la aplicación están configurados para aceptar únicamente archivos PDF por debajo de un umbral de tamaño predefinido. Esta es una medida de defensa crucial contra ataques de denegación de servicio (DoS) que intenten agotar el espacio en disco o los recursos de procesamiento mediante la subida de archivos masivos.
    - **Rol de orquestador, no de procesador:** El backend delega las tareas pesadas y potencialmente riesgosas (como el análisis de contenido

de un PDF) a los workers. Esto reduce su propia superficie de ataque y el tiempo que dedica a cada petición, haciéndolo más resiliente.

Almacenamiento de Objetos (MinIO)

- **Criticidad: Crítica.** Este componente es el repositorio de los activos más sensibles: los documentos originales y traducidos de los usuarios. Una brecha de seguridad que permitiera el acceso no autorizado a este servicio resultaría en una fuga directa de datos confidenciales.
- **Medidas específicas:**
    - **Aislamiento en red interna:** El servicio MinIO no está expuesto a Internet. Solo es accesible desde la red interna de Docker, lo que significa que únicamente otros contenedores autorizados (como el backend y los workers) pueden comunicarse con él.
    - **Acceso exclusivo mediante URLs prefirmadas:** Como se detalló anteriormente, no existe ningún mecanismo de acceso público a los ficheros. El único modo de acceder a un documento es a través de una URL temporal y firmada, generada bajo demanda por el backend, garantizando un control de acceso granular y efímero.

Worker de Procesamiento (Celery)

- **Criticidad: Alta.** El worker es el componente que manipula directamente el contenido de los documentos subidos por los usuarios y que alberga las credenciales para comunicarse con servicios externos. Es un punto crítico por dos razones: podría ser un objetivo para explotar vulnerabilidades en las librerías de procesamiento de documentos y gestiona secretos de la aplicación.
- **Medidas específicas:**
    - **Ejecución en un entorno aislado:** Gracias a la contenedorización con Docker, cada worker se ejecuta en un *sandbox* aislado. Si el procesamiento de un archivo PDF malicioso lograra explotar una

vulnerabilidad en una de las librerías subyacentes (ej. ReportLab, Pytesseract), el impacto estaría contenido dentro del contenedor, protegiendo al sistema anfitrión y a los otros servicios.

- **Gestión segura de la clave de API:** La clave para la API de OpenRouter se inyecta en el worker a través de variables de entorno, evitando que esté almacenada en el código fuente o en archivos de configuración dentro del control de versiones.

Dependencias Externas (API de OpenRouter)

- **Criticidad: Media.** Aunque es un servicio externo, representa un punto crítico en el flujo de datos. El contenido textual extraído de los documentos es enviado a este tercero para su traducción. La seguridad de esos datos, una vez abandonan la infraestructura propia, depende de las políticas y medidas del proveedor.
- **Medidas específicas:**
  - **Comunicación cifrada:** Toda la comunicación entre los workers y la API de OpenRouter se realiza exclusivamente a través de HTTPS, asegurando que los datos de texto estén cifrados durante su tránsito.
  - **Evaluación de políticas de proveedor:** Para un despliegue en producción, sería imperativo realizar una evaluación de la política de privacidad y seguridad de OpenRouter (o cualquier otro proveedor de LLM) para asegurar que sus prácticas de manejo de datos son compatibles con los requisitos de confidencialidad del servicio.

5. Aspectos legales y cumplimiento normativo

Además de las medidas técnicas de seguridad, el despliegue y operación de un sistema que procesa datos de usuarios exige el cumplimiento de un marco legal específico. En esta sección se exponen las normativas más relevantes consideradas en el diseño del proyecto y las implicaciones que tendrían en un entorno de producción.

5.1 Reglamento General de Protección de Datos (RGPD)

Dado que los documentos subidos por los usuarios pueden contener datos de carácter personal, el sistema está sujeto al Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, conocido como Reglamento General de Protección de Datos (RGPD).

- **Rol del responsable del tratamiento:** En el contexto de la aplicación, el operador del servicio (en este caso, el autor del proyecto) actuaría como **Responsable del Tratamiento**, siendo el encargado de garantizar la protección de los datos procesados.
- **Base jurídica para el tratamiento:** La base legal para el tratamiento de los datos es el **consentimiento explícito** del interesado. El acto de que un usuario seleccione voluntariamente un archivo, configure las opciones y pulse el botón para iniciar la traducción constituye una acción afirmativa e inequívoca, dando su consentimiento para el único fin de procesar y traducir dicho documento.
- **Derechos de los interesados:** El diseño del sistema facilita el cumplimiento de los derechos de los usuarios:
    - **Derecho de acceso:** Se garantiza al permitir que el usuario visualice y descargue tanto el documento original como el traducido.
    - **Derecho de supresión (Derecho al olvido):** La arquitectura de almacenamiento basada en task_id permite una implementación directa de este derecho. La eliminación de la carpeta asociada a

un task_id específico en el *bucket* de MinIO resultaría en el borrado completo y permanente de todos los artefactos relacionados con esa tarea (PDFs y metadatos JSON), sin dejar rastro de los datos procesados.

- **Medidas técnicas y organizativas:** Las estrategias detalladas en las secciones de este anexo (cifrado en tránsito y en reposo, control de acceso mediante URLs prefirmadas, aislamiento de datos y de servicios) constituyen las medidas técnicas implementadas para cumplir con los principios de seguridad y confidencialidad exigidos por el RGPD.

5.2 Propiedad intelectual de los documentos

Es fundamental clarificar la postura del sistema respecto a la propiedad intelectual del contenido. La aplicación actúa únicamente como una herramienta de procesamiento. No reclama, en ningún caso, derecho alguno sobre el contenido de los documentos subidos por los usuarios ni sobre el resultado de la traducción. La propiedad intelectual del material original y de la obra derivada (la traducción) pertenece íntegramente al usuario.

5.3 Políticas de servicios de terceros

El sistema depende de un servicio externo (la API de OpenRouter) para realizar la traducción. Esto implica una transferencia de datos (el contenido textual extraído) a un tercero. Por tanto, es necesario considerar los términos de servicio y la política de privacidad de dicho proveedor. En un despliegue público final de la aplicación, sería obligatorio informar a los usuarios de esta transferencia de datos y enlazar a las políticas correspondientes de OpenRouter, garantizando la transparencia sobre toda la cadena de tratamiento de datos.