Universidad Carlos III

Curso Desarrollo del Software 2024-25

Curso 2024-25

Entrega 2
Práctica Criptografía y Seguridad Informática

Fecha: **11/12/24**

Grupo: **85**

ID Prácticas: **20**

Alumnos: **MANUEL ANDRES TRUJILLO / 100423448 (100423448@alumnos.uc3m.es)**

VICENTE ANTONIO BARBATO / 100438114 (100438114@alumnos.uc3m.es)

Tabla de contenido

Gestor de Contraseñas

Propósito

En la actualidad, la seguridad de las contraseñas para acceder a aplicaciones ha adquirido una relevancia fundamental. Para garantizar un acceso seguro y evitar vulnerabilidades que puedan exponer nuestras credenciales, se recomienda utilizar contraseñas extensas y complejas. Además, es aconsejable emplear contraseñas únicas para cada aplicación en lugar de reutilizar la misma clave en diferentes plataformas. Sin embargo, esta práctica demanda un esfuerzo por nuestra parte, para poder recordar de forma precisa nuestras claves de acceso y puede dificultar la rapidez con la que accedemos a las aplicaciones. Por ello, hemos optado por desarrollar un gestor de contraseñas que no solo ofrece un entorno seguro y eficiente para almacenar claves de acceso, sino que también brinda recomendaciones para mejorar contraseñas que no cumplan con los estándares básicos de seguridad.

Generación de claves

Antes de entrar en el contexto de firma digital y certificados, cabe destacar que hemos implementado un esquema de cifrado híbrido. Este esquema nos permite derivar claves específicas para cada necesidad, reforzando así la seguridad global de nuestra aplicación.

Nuestro enfoque combina la robustez del cifrado asimétrico (RSA) con la eficiencia del cifrado simétrico (AES), lo que permite manejar de manera segura tanto las claves como los datos confidenciales. Cada método criptográfico aporta sus fortalezas, logrando un sistema más completo y seguro.

El proceso comienza con la generación de un par de claves asimétricas (clave pública y clave privada) utilizando el algoritmo RSA. Estas claves se almacenan en formato PEM y se protegen mediante cifrado avanzado. La clave privada se cifra con AES en modo GCM, utilizando una clave simétrica derivada de manera determinista mediante el uso de PBKDF2HMAC. Este derivador toma como entrada un "seed" y un salt generado dinámicamente, el cual incluye datos únicos del usuario y sus credenciales, lo que aumenta significativamente la seguridad al mitigar riesgos de predicción.

Durante el almacenamiento, la clave privada cifrada se guarda en un archivo que también contiene el salt, el vector de inicialización (IV), y el tag de autenticación generados durante el proceso de cifrado. Estos elementos son necesarios para el descifrado posterior. La clave pública, aunque no se cifra, también se almacena en un archivo separado en formato PEM, lo que asegura su integridad y disponibilidad.
Adicionalmente, la clave pública se utiliza para cifrar una clave simétrica que es derivada. Esta clave simétrica, una vez cifrada con RSA, se almacena junto con las claves asimétricas. Este esquema asegura que solo el poseedor de la clave privada pueda descifrar las claves simétricas y, por ende, acceder a los datos protegidos, garantizando la confidencialidad y la integridad del sistema.

Firma Digital

Hemos decidido llevar a cabo la aplicación de la firma digital como requisito de seguridad adicional, el cual se encargará de garantizar tanto la autenticidad como la integridad de los datos sensibles que se manipulan. En concreto, la implementación se utiliza para asegurar que las credenciales almacenadas dentro de nuestro gestor de contraseñas, no haya sino manipuladas y así confirmar que el origen de dichas credenciales es legítimo. Esto resulta crítico en un entorno de gestión de credenciales, donde la confianza en los datos es esencial para el correcto funcionamiento del sistema y la protección de la información del usuario.

Algoritmos implementados

En cuanto a los algoritmos, se ha optado por el uso de RSA debido a su probada robustez y su amplia adopción en sistemas criptográficos de clave pública. RSA permite la generación de firmas digitales confiables al tiempo que proporciona un nivel elevado de seguridad frente a ataques criptoanalíticos. El proceso de firma digital con RSA implica los siguientes pasos:

- Implementación de claves: Se hace uso de las claves genéricas anteriormente documentadas, de forma que la clave privada se utiliza exclusivamente para firmar los datos, mientras que la clave pública permite verificar la validez de la firma.

- Al firmar un mensaje o conjunto de datos, se calcula un hash criptográfico (SHA-256) del contenido, que actúa como una representación única y de tamaño fijo de los datos originales.

- El hash resultante se cifra con la clave privada utilizando RSA, generando la firma digital. Este cifrado asegura que solo el poseedor de la clave privada pueda crear la firma.

- Para la verificación, el receptor utiliza la clave pública del emisor para descifrar la firma y recuperar el hash original. Posteriormente, recalcula el hash de los datos recibidos y lo compara con el hash descifrado. Si ambos coinciden, la firma es válida y se confirma que los datos no han sido alterados.

Finalmente, cabe destacar, que en el caso de las firmas digitales no requieren almacenamiento persistente, ya que se generan y verifican dinámicamente durante el ciclo de vida de los datos protegidos.

Certificados

Generación de certificados

La generación de certificados de clave pública en la aplicación sigue un proceso estandarizado conforme a X.509. Primero, se genera un par de claves RSA compuesto por una clave privada y una clave pública, utilizando un tamaño de clave de 2048 bits.

Luego, se define el sujeto del certificado, si el certificado no es autofirmado, se crea una solicitud de firma de certificado (CSR), que incluye el sujeto y la clave pública, firmada con la clave privada del propietario. Posteriormente, el certificado se construye especificando al sujeto, al emisor, la clave pública, un número de serie único, en un periodo de validez, y extensiones.

El certificado se firma digitalmente con la clave privada del emisor utilizando el algoritmo SHA-256, y luego se serializa y guarda en formato PEM. Las claves privadas y los certificados se almacenan de forma segura en directorios específicos según su propósito.

Jerarquía de certificación

Llevamos a cabo el desarrollo de una jerarquía que permite la gestión de certificados emitidos de una forma eficiente, debido a que, demarcamos con claridad la separación de responsabilidades, protegiendo así la clave privada de la CA raíz minimizando su uso. Esto se logra al implementar un diseño de jerarquía de dos niveles, compuesta por CA raíz (AC1) y una CA subordinada, la cual se encarga de emitir y gestionar los certificados de los usuarios finales. Adicionalmente, esta jerarquía permite una gestión optima de los certificados emitidos, facilitando el control de revocaciones y validaciones dentro del sistema, y adaptándose a las mejores prácticas de infraestructura de clave pública PKI.

Decisiones tomadas

La implementación de la generación de certificados en la aplicación se llevó a cabo de forma cuidadosa para garantizar seguridad, organización y trazabilidad. Realizando, llevando a cabo una estructura funcional.

- Uso de la biblioteca cryptography: Se llevo a cabo la implementación de esta biblioteca en lugar de herramientas externas como OpenSSL para simplificar la integración y mantener el control del proceso dentro del código Python.

- Estructura de directorios: Optamos por adaptarnos a la estructura recomendada por la documentación proporcionada en aula globa, permitiendo un orden claro para cada autoridad certificadora, conformada por:

- Solicitudes: Almacena las solicitudes de firma de certificados (CSR) recibidas por la CA para ser procesadas.

- Crls: Contiene las listas de revocación de certificados (CRLs), que registran los certificados revocados por la CA.

- Nuevoscerts: Guarda los certificados emitidos por la CA, organizados para un fácil acceso y validación.

- Privado: Almacena las claves privadas de la CA de forma protegida, esenciales para firmar certificados y garantizar la autenticidad.

- Serialización y trazabilidad: Cada CA incluye un archivo de seguimiento (serial e index.txt) para registrar los números de serie y la validez de los certificados emitidos.

Implementación

Hemos llevado a cabo la implementación de certificados en distintos puntos críticos de la aplicación, garantizando la seguridad, autenticidad e integridad de las operaciones. Estas implementaciones son las siguientes:

1. Durante el registro de usuarios: Al registrar un nuevo usuario, se emite un certificado único asociado a su identidad. Este proceso asegura que cada usuario tenga un medio confiable de autenticación dentro del sistema, respaldado por la CA subordinada.

2. En la autenticación de usuarios: Durante el inicio de sesión, el sistema verifica el certificado del usuario para garantizar que fue emitido por una autoridad confiable y que no ha sido manipulado. Esta verificación incluye validar la firma del certificado y confirmar su vigencia, proporcionando una capa adicional de seguridad más allá de las credenciales tradicionales.

3. Control de revocación: Los certificados también se utilizan en el proceso de validación contra las listas de revocación (CRLs). Esto asegura que cualquier certificado comprometido o revocado no pueda ser utilizado dentro del sistema.

Complejidad y diseño

Diseño

El diseño de la aplicación sigue un enfoque modular y está orientado a objetos, lo que garantiza una estructura bien organizada y fácilmente mantenible. Este diseño se centra en encapsular responsabilidades específicas en módulos independientes, lo que facilita la escalabilidad y la incorporación de nuevas funcionalidades. La estructura global de la aplicación se desarrolla de la siguiente forma:

- Core: Este módulo se centra en implementar las operaciones esenciales relacionadas con la seguridad y la criptografía. Incluye funcionalidades como el cifrado y descifrado de datos, la generación y verificación de firmas digitales, la gestión de integridad mediante HMAC, y la interacción con la base de datos para garantizar un almacenamiento seguro de información sensible.

- Interfaz: Este módulo está dedicado a la interacción con el usuario. Se encarga de desarrollar las pantallas que permiten a los usuarios gestionar credenciales, iniciar sesión, registrarse, y realizar otras acciones dentro de la aplicación. Además, integra notificaciones visuales y validaciones para asegurar una experiencia de usuario fluida

- Gestión de claves: Este módulo se ocupa de la generación, almacenamiento y recuperación de claves criptográficas. Las claves privadas se protegen mediante cifrado avanzado para evitar accesos no autorizados, mientras que las claves públicas se gestionan para su uso en operaciones de verificación y autenticación.

Por otro lado, se lleva a cabo un desarrollo de interfaz minimalista y sencillo, enfocado a la comodidad y facilidad de uso al usuario, de esta forma se consigue que sea una experiencia agradable e intuitiva.

Complejidad

La complejidad y rendimiento del sistema, se ve altamente influenciado por las operaciones criptográficas y de gestión de datos, por lo que es necesario encontrar un equilibrio entre seguridad y eficiencia, ya que, por norma general, si implementamos muchas capas de seguridad o algoritmos con costes altos computacionalmente hablando, el rendimiento de nuestra aplicación se verá afectado de forma negativa significativamente. Por otro lado, no podemos permitirnos realizar una seguridad trivial, de forma que nuestra aplicación mantenga el mejor rendimiento posible

- Cifrado y Descifrado (AES): Estas operaciones tienen una complejidad lineal $O(n)$, donde n es el tamaño de los datos procesados. Este enfoque asegura que los datos sean protegidos eficientemente durante las operaciones de almacenamiento y recuperación.

- Derivación de Claves (PBKDF2HMAC): La generación de claves tiene una complejidad de $O(c \cdot n)$, donde c es el número de iteraciones configuradas, en el caso de nuestra implementación, c toma el valor de 100.000 y n es la longitud

de la clave. Esta operación es intensiva computacionalmente, pero necesaria para garantizar la robustez frente a ataques de fuerza bruta.

- Firma y Verificación Digital (RSA):
    - Generar una firma digital tiene una complejidad de O(k3), debido a las operaciones de exponenciación modular con claves privadas.
    - Verificar una firma tiene una complejidad de O(k2), lo que es más eficiente y adecuado para procesos de autenticación frecuentes.
- Base de Datos: Las consultas e inserciones en la base de datos tienen una complejidad promedio de O (log m), donde m es el número de registros almacenados. Esta eficiencia permite un acceso rápido a los datos incluso con un número elevado de usuarios.

Mejoras Implementadas

Almacenamiento en Base de Datos:

Se utiliza una base de datos para almacenar de forma segura las credenciales y otros datos sensibles. Las contraseñas se almacenan como hashes generados con el algoritmo bcrypt, asegurando resistencia frente a ataques de fuerza bruta. Por otro lado las credenciales de los usuarios han sido almacenadas, mediante la implementación de un cifrado híbrido, llevando a cabo el siguiente flujo de trabajo:

- Se usa cifrado simétrico (AES) para las credenciales.
- Se aprovecha una clave privada RSA para derivar la clave simétrica.
- Se refuerza la seguridad con HMACs y firmas digitales.

Validación de Datos de Usuario:

1. Contraseña:
    - Longitud mínima de 8 caracteres.
    - Inclusión de al menos una letra mayúscula, una letra minúscula y un carácter especial.
2. Correo Electrónico:
    - Validación de formato mediante una expresión regular.
    - Verificación de coincidencia entre el correo ingresado y su confirmación.

Este enfoque minimiza errores de entrada y garantiza un nivel básico de seguridad en las credenciales.

Captura de Excepciones con Logs:

Todas las operaciones críticas, incluidas las relacionadas con criptografía, se registran mediante un sistema de logging. Esto facilita el monitoreo de eventos, la trazabilidad y la identificación de problemas en tiempo real.

Al registrar un nuevo usuario, se observará confirmación del proceso llevado a cabo

```
Certificado emitido y almacenado para el usuario 'Usuario Prueba'.
El certificado del usuario 'Usuario Prueba' es válido.
Verification token sent successfully
```

Al realizar el registro de una nueva credencial

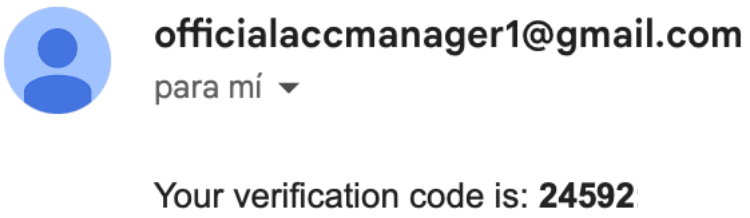
```
[INFO   ] [Encrypted username cifrado] tyYDmCI6SzpcBanQULqQsoG+Y/wZoMEL38tp0im9a7A=
[INFO   ] [Encrypted password cifrado] myDveUAd2/m2ZByqQWqbxZefPKHZR7IJO89TT3up6Ow=
[INFO   ] [HMAC generado] c56f4f77414ccf96e7a43564ffffc8a1d0e5154df9e3f2fcb56bac5c8b038eaa
[INFO   ] [HMAC generado] 1adde4190e6a30061683f639790df7c6907e6d76d1a5aa67f7cdbfde68b6a744
[INFO   ] [Digital Signature] 6912a5ddc8827730a152d29b2a97f428f1bac62f4be1cddbea75a9cca77ce1a0ada3f7c1f9bb5333b9d4deee8a50defd8ebf18921c63c4475b3f232eadac91a22f73b9501a636899
[INFO   ] ------------------------------
```

Al visualizar credenciales

```
[INFO   ] [Verificación HMAC para Username] exitosa
[INFO   ] [Verificación HMAC para Password] exitosa
[INFO   ] [Username descifrado] G123
[INFO   ] [Password descifrado] G123
[INFO   ] [Digital Signature] La firma es válida.
[INFO   ] ------------------------------
```

Autenticación de Doble Factor (2FA):

Implementación de un sistema de doble factor basado en correos electrónicos. Al intentar acceder, el usuario recibe un token temporal (de cinco dígitos) que expira en 60 segundos, añadiendo una capa adicional de seguridad frente a accesos no autorizados.

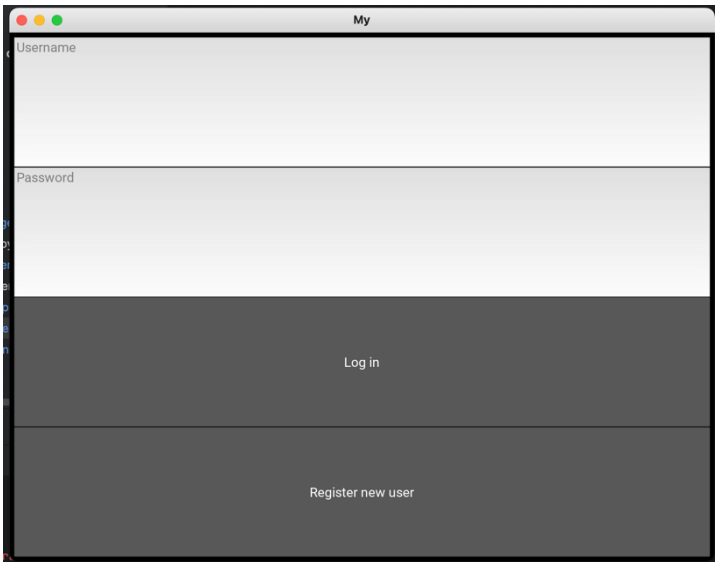


Interfaz gráfica:

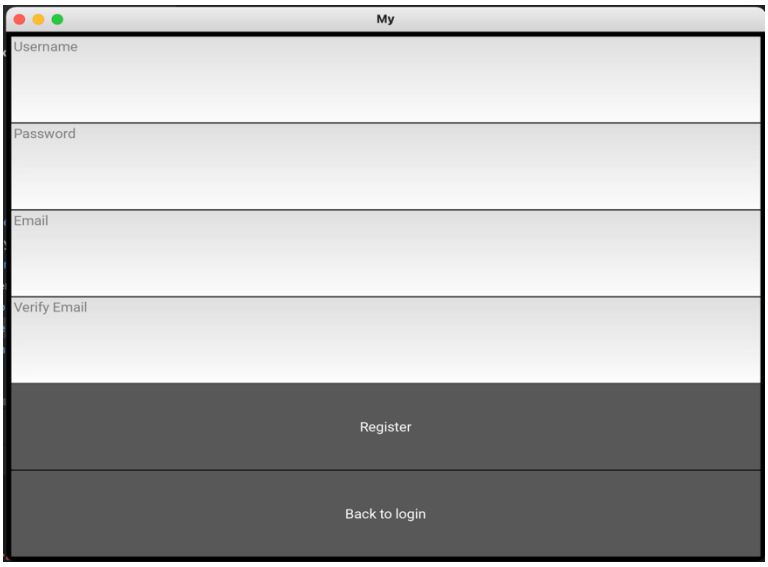
Hemo realizado la implementación de una interfaz grafica fácil e intuitiva de utilizar, mejorando la experiencia de usuario de forma notoria al hacer uso de nuestro gestor de contraseñas. La implementación se llevó a cabo mediante el uso de la

librería de Python llamada Kivi, ya que se adecua perfectamente a las necesidades de nuestro gestor y sin necesidad de gran esfuerzo ni complicaciones para su ejecución.

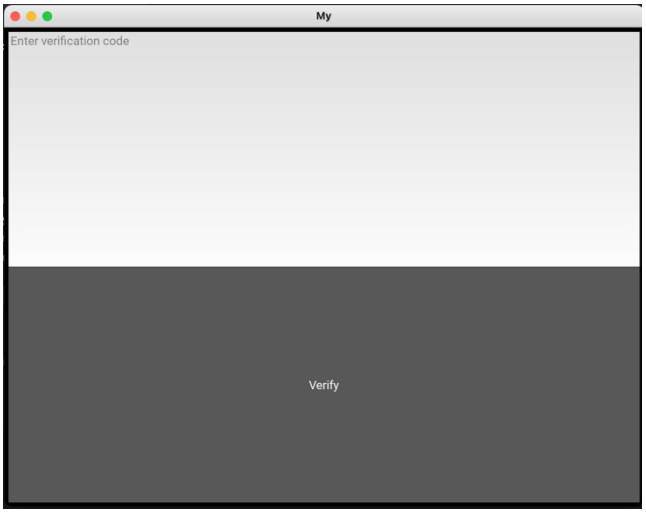
Pantalla de Inicio



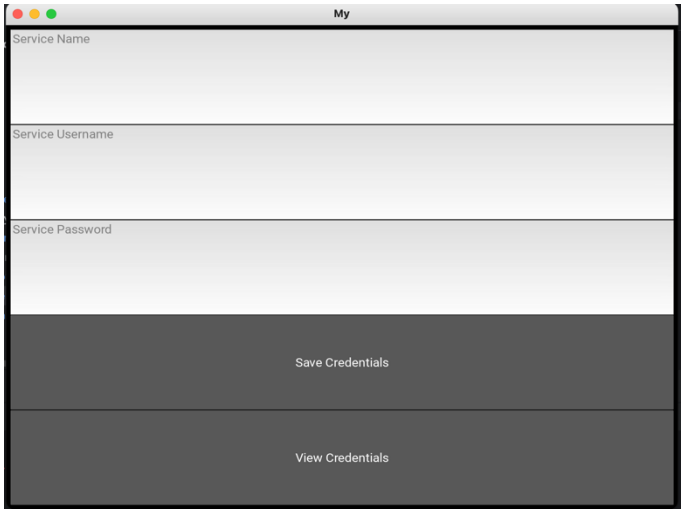
Pantalla de registro



Verificación 2FA



Registro de credenciales



Visualización de credenciales