Criptografía y seguridad informática

Entrega 2

GRUPO - 8305

Alberto Díaz-Pacheco Corrales - 100451158

Jordi Pérez Pastor - 100429127

Curso 2022/2023

Grado en Ingeniería Informática

Grupo - 8305
Alberto Díaz-Pacheco Corrales
Jordi Pérez Pastor

1. ¿Cuál es el propósito de su aplicación?

La aplicación **Theowall** es un **gestor de contraseñas**. Permite poder guardar las credenciales del usuario de cualquier red social, página web, aplicación de forma segura, facilitando que el usuario pueda poner contraseñas más y complejas seguras sin miedo a que se le olviden.

Por el momento el usuario puede **registrarse**, **acceder** a la aplicación, **crear** credenciales, **modificar** credenciales, **eliminar** credenciales y modificar los datos de **perfil del usuario**.

La aplicación funciona con una base de datos con archivo **.json**. En dicho archivo se guardan los datos del perfil de usuario junto con el hash de la contraseña de acceso a la app y el contenido (las credenciales) cifrado al cual solo el usuario puede acceder con su **contraseña maestra** (la contraseña de acceso a la aplicación).

Además, la aplicación cuenta con un generador de documentos. Dichos documentos contienen las credenciales del usuario solicitante. Este documento está **firmado** por la aplicación. Esta firma se puede **verificar** desde la propia aplicación.

2. ¿Para qué utiliza la firma digital? ¿Qué algoritmos ha utilizado y por qué? ¿Cómo gestiona las claves? ¿Cuál es la PKI que ha desarrollado?

En la aplicación hay una funcionalidad que genera un documento con las credenciales del usuario. Este documento lo firma la aplicación. El algoritmo de firma digital que se ha usado es RSA con tamaño 2048. No se ha usado el tamaño 1024 ya que no es del todo seguro. Tampoco se ha usado 4096 ya que se ha considerado no necesario para el caso concreto de la práctica siendo RSA-2048 suficientemente seguro y más rápido que RSA-4096.

Las claves se guardan en formato PEM. En concreto, la clave privada está cifrada con una contraseña. Esto significa que en el archivo “`keys/private_key.pem`” la clave privada no está en claro, sino cifrada. Para descifrarla hace falta la contraseña.

En este caso, la contraseña de cifrado de la clave privada se encuentra en el propio código. Sabemos que no debería estar ahí, pero por simplicidad se nos ha permitido hacerlo de esta manera.

No se ha desarrollado ningún PKI para el proyecto.

3. ¿Qué tipo de autenticación ha implementado? ¿Por qué ha escogido ese tipo y no otro? ¿Cómo lo ha implementado?

Como se comentó en la anterior entrega, para el proceso de autenticación se ha hecho uso de las funciones hash. El hash se utiliza para guardar la contraseña maestra del usuario. Cuando un nuevo usuario se registra, se realiza la función hash de la contraseña y se guarda en la base de datos, de tal manera que siempre que quiera acceder a la aplicación con su usuario y contraseña se realiza la función resumen de la contraseña que el usuario teclea y se compara con el hash guardado en la base de datos. Si coinciden, el usuario accede, sino, se imprime en pantalla un error de “contraseña incorrecta”.

Se ha usado SHA-256 ya que SHA-128 no es seguro. SHA-256 es suficientemente seguro.

```
def hash_pwd(pwd):
    "Recibe la contraseña y devuelve el hash-SHA256"
    pwd_b = bytes(pwd, 'utf-8')
    hash = hashlib.sha256()
    hash.update(pwd_b)
    pwd_h = hash.hexdigest()
    return pwd_h
```

4. Si ha realizado mejoras, explique cuáles y las implicaciones de seguridad de cada una de ellas en su programa/aplicación

Además de todos los aspectos anteriores, también se ha tratado el tema de las validaciones de los parámetros editables por el usuario, en concreto sus datos personales en el perfil de usuario y a la hora de registrarse.

También hay un control de errores. En los errores previstos, se manejan imprimiendo un mensaje de error explicativo. Por ejemplo, si el usuario escribe una dirección de email inválida, se imprime en pantalla el siguiente error:

```
+ ERROR --> LA DIRRECIÓN EMAIL NO ES VÁLIDA
```

O si se ha hecho un ataque a la base de datos y se ha modificado:

```
+ ERROR --> La base de datos ha sido dañada
```

Para los errores que no están contemplados en el código, se imprime un mensaje de error en pantalla terminando el programa:

```
exceptions.Exceptions: ⚠ HA OCURRIDO UN ERROR ⚠
```

Como se puede ver en la captura, este error lo maneja la clase Exceptions. Esto permite avisar de que ha ocurrido un error, pero no da pistas sobre qué es lo que ha pasado. Así evitamos que falsos usuarios o usuarios fraudulentos encuentren fácilmente una vía de ataque.

También se ha implementado una mejora en el sistema de registro. Cuando un usuario nuevo se registra, se le envía automáticamente un correo electrónico con un código de verificación a la dirección que el usuario ha introducido. El usuario debe introducir en la aplicación el código recibido por email para poder finalizar el registro. Esto previene un ataque de fuerza bruta y también impide que la dirección de correo electrónico sea falsa.