Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

PAI 5 - MOBIFIRMA. DESARROLLO DE PILOTO PARA COMPRAS CON DISPOSTIVOS MÓVILES PARA UNA CADENA HOTELERA

Grado en Ingeniería Informática – Ingeniería del Software

Seguridad en Sistemas Informáticos e Internet

Curso 2023 – 2024

Participantes:

Juan Luis Ruano Muriedas

José Joaquin Rojas Romero

Antonio José Suárez García

Índice

Resumen ejecutivo

Se ha realizado un fichero para la generación de **claves RSA seguras** para los empleados y el registro de las mismas en la base de datos y se ha utilizado **OpenSSL** para la generación del **certificado digital** para la conexión segura.

En la parte del **cliente**, como se especificaba, es una aplicación de *Android Studio* donde se recogen los parámetros del pedido y la identificación del empleado que provee su clave para f**irmar la petición** y conectar de forma segura con **TLSv1.3 con el servidor.**

En la parte del **servidor**, se han realizado métodos para la generación de ***logs*** mensuales, el ***KPI*** y la conexión segura con los clientes mediante **TLSv1.3**, así como las comprobaciones pertinentes de los datos recibidos y su manejo, ya sea cuando son correctos o incorrectos.

Generación de claves

Para la **generación de claves seguras** se ha creado el fichero ***new_key_generator.py***. En este fichero se han definido las funciones necesarias: *randomEmployeeID()*, *startDatabase()*, *insertKey_PUB()*, *generateKey()* y *saveKeys()*.

- La función ***randomEmployeeID()*** crea IDs para nuevos trabajadores. Simplemente genera un número aleatorio de 5 dígitos.
- La función ***startDatabase()*** crea la base de datos SQLite con la tabla *worker*, en la cual se almacenan las IDs y las claves públicas de los trabajadores de la empresa.
- La función ***insertKey_PUB()*** almacena la ID y la clave pública de cualquier trabajador a la base de datos.
- La función ***generateKey()*** genera el par de claves RSA de los trabajadores. Genera un par de claves RSA para cada trabajador.
- La función ***saveKeys()*** guarda la clave privada del trabajador en un archivo ***KEY-****IDtrabajador****.pem*** para que pueda ser repartida mediante un canal seguro al trabajador.
- Además, se ha generado un certificado para la aplicación usando ***OpenSSL***.

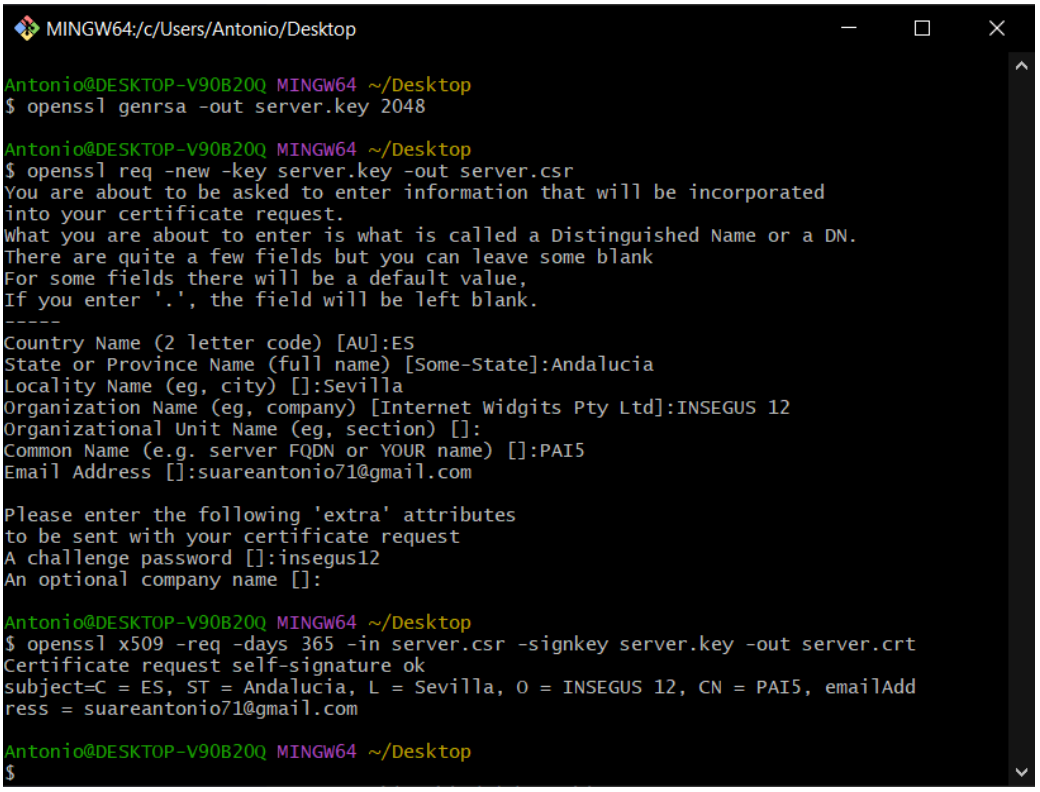

Figura 1. Método de generación del certificado digital de la aplicación.

Cliente de la aplicación

La aplicación se ha desarrollado mediante el lenguaje de programación ***Java*** a través del framework ***Android Studio*** y presenta la siguiente interfaz:

Figura 2. Interfaz de la aplicación móvil

Hemos considerado que las camas, sillas, sillones y toallas son los elementos necesarios que se pedirán para el reabastecimiento de los hoteles. Y por último se incluye el **ID del trabajador** que va a hacer el pedido. En cuanto a la lógica de la aplicación, como el cliente pidió, la magnitud de las cantidades a pedir tendrá un **límite de entre 0 y 300 unidades** y al pedir el identificador del trabajador la aplicación hará uso de las **claves RSA** que tenga almacenadas del trabajador generadas por el servidor anteriormente para posteriormente **firmar la petición** con el **algoritmo SHA256 y la clave privada.**

A continuación, la aplicación conectará de forma segura a través de **TLSv1.3 con el socket** del servidor donde se almacenan los datos y se **envía el mensaje, la id y la firma** del empleado codificada en **base64** al servidor que se encargará de la **verificación de la firma.** En caso de que necesite hacer pruebas, la aplicación entregada el cliente de **ID 58883** es el que posee la **firma digital** y por tanto otros ID no funcionarán hasta que se incluyan.

Servidor de la aplicación

Para el servidor de la aplicación se ha utilizado ***Python*** por ser un lenguaje de alto nivel con buena compatibilidad de ***SQLite3*** como se pedía en las especificaciones y se ha creado en el fichero ***hotel_server.py***. Este archivo constituye un servidor que administra pedidos de objetos por parte de los trabajadores de una empresa, implementando medidas de seguridad y registro de actividad. Permíteme desglosar sus principales funciones y estructura:

Inicialización de la base de datos

La función ***setUpDatabase(self)*** crea o conecta a una base de datos **SQLite** llamada *database.sqlite3* si no existe, y crea dos tablas: *employee* para almacenar datos de empleados y sus **claves RSA**, y *order_details* para registrar los detalles de los pedidos.

Funciones de registro de actividad:

addEntryLog(self, msg)*:** Registra eventos en un archivo de registro diario, incluyendo la fecha y hora del evento. En el caso de crear un nuevo *.log,* es decir, al pasar al mes siguiente, se llama a la función ***generarKpi(self) para calcular el kpi del mes que acaba de terminar.

```
1  FALLO, Se ha recibido un mensaje de un usuario no verificado Día: 09/04/2024
2  FALLO, El mensaje del usuario 123 ha sido corrompido. Día: 09/04/2024
3  EXITO, El trabajador 123ha solicitado: 1 camas, 1 toallas, 1 sillas, 1 sillones Día: 09/04/2024
4  EXITO, El trabajador 123ha solicitado: 13 camas, 13 toallas, 133 sillas, 13 sillones Día: 09/04/2024
5
```

Figura 3. Ejemplo de formato de log con algunas entradas.

***generatekpi(self)*:** Lee los logs de los dos meses anteriores y del mes que acaba de terminar y se calcula el ratio de los tres y la tendencia del nuevo mes y se escribe en el archivo ***kpi.log*.**

kpi.log

```
1  2024-01, ratio: 50.0, tendency: 0
2  2024-02, ratio: 50.0, tendency: 0
3  2024-03, ratio: 50.0, tendency: 0
4  2024-04, ratio: 50.0, tendency: +
5
```

Figura 4. Ejemplo de formato de kpi con algunas entradas. Las dos primeras son 0 como se indicaba y a partir de ahí se calcula la tendencia y el ratio.

Funciones de verificación y procesamiento de mensajes

***signatureVerification(self, signature, pbk, msg, employeeID)*:** Verifica la firma digital de un mensaje utilizando la clave pública RSA proporcionada, comparando el hash del mensaje con la firma proporcionada.

***checkMsg(self, bed_number, towel_number, chair_number, armch_number, employeeID)*:** Verifica si un trabajador ha realizado un número excesivo de pedidos en las últimas 4 horas, limitando a tres pedidos como máximo.

***detectNonVerifiedEmployee(self, employeeID, hashedMessage, message)*:** Comprueba la validez de los datos del trabajador utilizando su **clave pública RSA** para verificar la firma digital del mensaje.

***insertData(self, bed_number, towel_number, chair_number, armch_number, employeeID)*:** Inserta los detalles del pedido en la base de datos.

Función principal y ejecución del Servidor

***serverRun(self)*:** Inicia un **servidor SSL/TLS** en el puerto especificado (por defecto, 7070). Escucha las conexiones entrantes, procesa los mensajes recibidos y responde según la validez de los datos y el cumplimiento de los límites de solicitud. La IP utilizada es *192.168.1.32* para las pruebas en local y deberá ser sustituida por la dirección IP que el cliente le quiera poner a su servidor.

La conexión se abre de forma segura con el cliente con el protocolo **TLSv1.3** donde el servidor especifica que solo se utilice esta versión del protocolo para garantizar la máxima seguridad.

Para poder lanzar el servidor es necesario abrir un entorno virtual de python con el proyecto importado dentro e instalar todo los requisitos del archivo requirements con ***pip install -r requirements.txt*** y por último sólo necesita ejecutar el archivo ***hotel_server.py*** y el servidor empezará a captar peticiones del cliente. Tenga en cuenta que en este modelo de desarrollo el servidor no se detendrá hasta que se termine el proceso.

Conclusión

Con el sistema desarrollado, un empleado del hotel con un ID previamente asignado, será capaz de realizar peticiones de recursos y materiales al servidor proveedor de dichos recursos de una manera segunda, confidencial e íntegra, quedando registrados todos los posibles errores o éxitos en el proceso en el propio servidor.

Bibliografía

TLS/SSL con Python. https://docs.python.org/3/library/ssl.html

Restringir TLSv1 y TLSv2.
https://stackoverflow.com/questions/57599480/python-requests-use-tls-v-1-3

Framework Android Studio para el desarrollo de aplicaciones para móvil.
https://developer.android.com/studio?hl=es-419

SQLite3 para Python.
https://docs.python.org/3/library/sqlite3.html

SQLite para Windows.
https://www.sqlite.org/index.html

Esquemas de generación de claves y firma otorgados por la plataforma de la enseñanza.
https://ev.us.es/bbcswebdav/pid-4272712-dt-content-rid-53899296_1/xid-53899296_1

Openssl proyecto oficial.
https://www.openssl.org/

Cómo configurar las redes de Android Emulator
https://developer.android.com/studio/run/emulator-networking?hl=es-419