MEMORIA Seguridad y Api Rest - Art With Drones

INFORME DE DESARROLLO

AD_Registry

Se ha implementado un sistema de contraseñas con cifrado irreversible. Cada dron creará su propia contraseña a la hora de registrarse por primera vez, esta se almacenará en la BBDD de Registry/Engine cifrada mediante **SHA256** (de manera irreversible). Esta contraseña se almacena en el archivo de cada dron (al que solo él tendría acceso) junto con su ID.
La comunicación entre Registry y Dron se realiza a través de **Api Rest**, de manera que Registry expone la Api que es consumida por cada Dron. En cuanto al cifrado, la propia tecnología de https (se dispone de un certificado autofirmado) dispone de cifrado asimétrico.

AD_Engine:

Para la autenticación de los Drones, se ha mantenido el sistema de Sockets.
Se ha agregado un sistema de tokens caducables (20 segundos), un cifrado híbrido mediantes RSA + AES para el envío de los mapas y un cifrado asimétrico mediante RSA (con key_size de 4096 bytes) para el resto de la comunicaciones.
A continuación se detallan cada unos de ellos:

-**Tokens caducables:** los tokens proporcionados por Registry tendrán una caducidad de 20 segundos (se almacena un timestamp junto al token), si pasado ese tiempo el Dron no se ha autentificado, deberá solicitar un nuevo token.
-**Cifrado Híbrido:** dado al peso que conlleva enviar el mapa, se ha optado por implementar un sistema que cifra el propio mapa por cifrado simétrico (AES) y después cifra dicha key simétrica con cifrado asimétrico (RSA). Esto hace posible el funcionamiento del programa, debido a las limitaciones de RSA (key_size debería ser muchísimo mayor para permitir eso).
-Cifrado Asimétrico: para el resto de comunicaciones se ha optado por un cifrado asimétrico. El intercambio de claves se realiza tras la autentificación del dron y estas se almacenan en archivos .pem individuales para cada aplicación.

Para controlar imprevistos climáticos se ha implementado un consumo del api de **OpenWeather**. Reemplazando este a “AD_Weather” que deja de existir.
Para seleccionar la ciudad tan solo hay que escribir el nombre de esta en el fichero “ciudad.txt”.

Por último se ha implementado un sistema de **auditorías** que registra todas las acciones relevantes durante la ejecución en un fichero .log de manera estructura. De estas acciones se conocerá: ip del protagonista, momento exacto en el que ocurre, tipo de acción/ incidencia (registro, recepción, envío….) y una descripción de la misma.

AD_Dron:
Como ya se ha mencionado, el dron debe seleccionar su propia contraseña que sólo él conocerá/almacenará. Deberá iniciar sesión en el Registry para recibir su token caducable con el que se identificará en el Engine.

DESPLIEGUE

Ejecutar Kafka:
docker-compose up(en termina en la carpeta *prod_cons*)

Ejecutar AD_REGISTRY:
python AD_Registry.py
Después introducir contraseña (090803)

Ejecutar AD_ENGINE:
python AD_Engine.py <Puerto de escucha> <Máximo de drones> <IP del Broker> <Puerto del Broker> <IP del AD_Weather> <Puerto del AD_Weather> '-r'
Ejemplo:
python AD_Dron.py 0.0.0.0 8002 localhost 29092 0 0.0.0.0 8000
Si se usa -r, se recuperar el estado en el que engine se desconectó.

Ejecutar AD_DRON:
python dron.py <Engine IP> <Engine Port> <Brocker IP> <Brocker Port> <Dron ID> <Registry_ip> <registry_port>
Aclaración: Para crear nuevos drones, usar IP = 0 como en el ejemplo
Ejemplo:
python AD_Engine.py 8002 10 localhost 29092 0.0.0.0 8001

Ejecutar FRONT:
node front_https.js

CORRECCIONES DE LA ENTREGA ANTERIOR

Engine puede perder la conexión y volver a conectarse, todos continúan por donde iba, para ello solo hay que añadir -r al final de la ejecución de AD_Engine.
El comportamiento del sistema para abordar temperaturas extremas ya funciona como debería, mientras esté fuera del umbral los drones volverán al inicio, cuando vuelva al umbral retoman la figura.
Los movimientos de los drones ya son fluidos y no dará error si registramos 10 drones para una figura compuesta por 8.