ADR - Seguridad - Autenticacion Sistema - Sistema

Estado	VIGENTE
Partes interesadas	AREA GOBIERNO TECNOLOGICO
TAGS	AUTENTICACION SEGURIDAD
Tomada el	24 abr 2024
Documentación Asociada	

Fichero	Modificado
Archivo PNG image-2024-2-15_13-55-28.png	19/09/2024 by CARLOS GONZALEZ SANTIAGO
Archivo PNG image-2024-2-15_13-38-8.png	19/09/2024 by CARLOS GONZALEZ SANTIAGO
Archivo PNG image-2024-2-15_13-54-3.png	19/09/2024 by CARLOS GONZALEZ SANTIAGO
Archivo PNG image-2024-2-15_14-1-39.png	19/09/2024 by CARLOS GONZALEZ SANTIAGO
Archivo PNG image-2024-2-15_13-48-55.png	19/09/2024 by CARLOS GONZALEZ SANTIAGO

Descargar todo

Solución Afectada	TODAS

Responsable	Aprobado por	Consultados	Informados
LUIS MARTINEZ FONTIVEROS ALBERTO FUENTES GOMEZ	LUIS MARTINEZ FONTIVEROS		AREA GOBIERNO TECNOLOGICO DESARROLLO SOFTWARE ECONOMICO-FINANCIERO (NTTDATA) DESARROLLO SOFTWARE ASISTENCIAL (NTTDATA) DESARROLLO SOFTWARE RRHH (AYESA)

Asunto a Decidir	Describe brevemente la decisión arquitectónica tomada. Se requiere decidir el mecanismo de autenticación y/o autorización para la comunicación entre sistemas del SAS teniendo en cuenta los requisitos minimos de seguridad exigibles en el SAS a un sistema de registro de actividad con validez legal.

Identificación de la decisión	Describe brevemente la decisión arquitectónica tomada. **Consideramos que la alternativa 3** es la que presenta un mejor balance coste beneficio, y que nos mantiene un modelo claro y sencillo, con capacidades de administración de los clientes autorizados a verter auditoria, manteniéndose además dentro de los estándares, y con un impacto en costes muy reducido tanto evidentes en MVP como ocultos cuando se inicie el proceso de integración en las aplicaciones terceras. **Keycloak ya proporciona una solución para el problema planteado** esta solución se apoya en un estándar como OAuth2.0. Al usar la misma herramienta y los mismos recursos que para el resto de autentificaciones **evitamos introducir un sistema adicional, duplicidades, el aumento de la complejidad y solapamientos con los sobrecostes que conlleva**. mantenemos el sistema simple **El uso de un clientId con un secret** asociado es en gran medida equivalente a la solución de un API-KEY dónde usamos un ID del cliente y una secret para obtener un token autentificado seguro con los roles y autorizaciones que le corresponda. Desde el punto de vista de la máquina invocante varia ligeramente el flujo ya que invoca a la URL proporcionada con el clientId y el password y obtiene el token generado por keycloak, a partir de este punto el funcionamiento/flujo es el estándar para cualquier autentificación por Oauth2.0 Al seguir la estrategia **con Keycloack no es necesario tener diferentes estrategias de seguridad. por ejemplo un servicio invocado por un usuario y por una máquina no necesita una autentificación OAUTH y apy-key, le basta con OAUTH.** **Keycloak es altamente configurable** y adaptable permitiéndonos adaptar la seguridad al nivel exacto que se requiera para cada cliente; es mas completo y flexible. **Una vez establecida la configuración deseada se puede generar un JSON con dicha configuración e importarlo para los nuevos clientes, convirtiendo la creación de nuevos clientes en algo trivial.** Al mantenernos en la opción de keycloak obtenemos una solución cercana a API-KEY en sencillez, con facilidad para operarlo e integrarlo en DEVOPS.

Contexto	Con la proliferación de la arquitectura de referencia y los desarrollos basados en microservicios unida a la estrategia de autenticación y autorización centralizada (SSO) de la organización. Por lo tanto, se solicita una propuesta de mecanismo de autenticación y autorización que sea adecuado para asegurar la comunicación entre sistemas.

Alternativas Consideradas

Se analiza el uso de API Keys y de Client Secret como mecanismo de autenticación y autorización para la comunicación entre sistemas del SAS. Esta propuesta se divide en tres alternativas:

Alternativa 1:

Api key para todos los sistemas igual. Guardar el valor en k8s y obtenerlo desde una variable de entorno. Es sólo autenticación, no incluye autorización (es decir, cualquier sistema puede logar cualquier evento)

Alternativa 2:

Incluir una autorización por cada sistema productor. En este caso habría que disponer de una persistencia que almacenara API-KEY, sistema y eventos autorizados a logar. Esta alternativa es más elegante aunque más costosa, y permite un grano más fino de autenticación y autorización.

Alternativa 3:

Otra alternativa al Ali-key es el uso de Keycloak con Client Id y secret. Este mecanismo permite la autentificación y autorización entre máquinas.

Alternativa 1

Las API Keys permiten autenticar y autorizar a un sistema tercero a acceder al recurso del sistema responsable del mismo. Para ello, cada sistema tercero debe obtener una API Key y una API Secret Key.

Posibilidades

- **Autenticación y autorización:** Las API Keys permiten autenticar y autorizar a un sistema tercero a acceder al recurso del sistema responsable del mismo.
- **Autorización:** Las API Keys permiten autorizar a un sistema tercero a acceder al recurso del sistema responsable.
- **Flexibilidad:** Las API Keys pueden ser utilizadas para restringir el acceso a al recurso del sistema en función de las necesidades de cada sistema tercero.
- **Seguridad:** Las API Keys pueden ser utilizadas para proteger el recurso del sistema de accesos no autorizados.

Ventajas

- **Fácil implementación:** Las API Keys son fáciles de implementar y utilizar.
- **Escalabilidad:** Las API Keys son escalables y pueden ser utilizadas para gestionar un gran número de sistemas terceros.
- **Coste:** Las API Keys son una solución de bajo coste.

Desventajas

- **Requiere gestión:** Las API Keys requieren una gestión adecuada para evitar que sean utilizadas de forma fraudulenta.
- **No es adecuada para todos los casos:** Las API Keys no son adecuadas para casos en los que se requiere una autenticación y autorización más compleja, como la autenticación multifactor.
- **Nivel de seguridad muy básico**: La API Key no puede ser revocada fácilmente sin altos costes, dado que es única cualquier sistema que la conozca puede hacer uso de ella para registrar auditoria, y el control es prácticamente inexistente una vez esté entregada al primer proyecto.

Coste Adopción

Dentro de las ventajas por parte de se plantea que este sistema es de bajo coste pero no se está contando con la duplicidad que implica la multiplicidad en los sistemas de autenticación, así pues la evaluación de coste que se hace al respecto es la siguiente:

- Coste de API Keys
- Coste Integración con Autentificación Keycloak (la autentificación máquina - máquina no es excluyente con usuario - máquina por lo que el sistema destino debe soportar ambas autentificaciones)
- Coste por solapamiento. Estaríamos tratando la autentificación con dos sistemas diferentes con lo que tenemos duplicidades que complican la operación.
- Coste de operación (Es necesario catalogar y administrar los api-key, esta es una tarea fundamentalmente manual)

Así pues el coste sería:

- **Coste de API Keys Coste bajo**
- **Coste Integración** con Autentificación Keycloak **Coste medio** (aumenta la complejidad del software en caso de ser necesaria)
- **Coste por solapamiento**. **Coste alto**
- **Coste de operación Coste medio**

Alternativa 2

Realmente esta alternativa es una expansión de la alternativa 1. en esta alternativa plantea lo siguiente "Incluir una autorización por cada sistema productor. En este caso habría que disponer de una persistencia que almacenara API-KEY, sistema y eventos autorizados a logar. Esta alternativa es más elegante aunque más costosa, y permite un grano más fino de autenticación y autorización."

Ventajas

- **Fácil implementación:** Las API Keys son fáciles de implementar y utilizar.
- **Escalabilidad:** Las API Keys son escalables y pueden ser utilizadas para gestionar un gran número de sistemas terceros.
- **Coste:** Las API Keys son una solución de bajo coste.
- **Grano fino de autorización**

Desventajas

- **Requiere gestión:** Las API Keys requieren una gestión adecuada para evitar que sean utilizadas de forma fraudulenta.
- **No es adecuada para todos los casos:** Las API Keys no son adecuadas para casos en los que se requiere una autenticación y autorización más compleja, como la autenticación multifactor.
- **Nivel de seguridad muy básico**: La API Key no puede ser revocada fácilmente sin altos costes, dado que es única cualquier sistema que la conozca puede hacer uso de ella para registrar auditoria, y el control es prácticamente inexistente una vez esté entregada al primer proyecto.
- **Requiere de un software adicional** para poder gestionar adecuadamente todas las API-Keys

Coste Adopción

Dentro de las ventajas por parte de se plantea que este sistema es de bajo coste pero no se está contando con la duplicidad que implica la multiplicidad en los sistemas de autenticación, así pues la evaluación de coste que se hace al respecto es la siguiente:

- Coste de API Keys
- Coste Integración con Autentificación Keycloak (la autentificación máquina - máquina no es excluyente con usuario - máquina por lo que el sistema destino debe soportar ambas autentificaciones)
- Coste por solapamiento. Estaríamos tratando la autentificación con dos sistemas diferentes con lo que tenemos duplicidades que complican la operación.
- Coste de operación(Es necesario catalogar y administrar los api-key, esta es una tarea fundamentalmente manual)

Así pues el coste sería:

- **Coste de API Keys Coste bajo**
- **Coste Integración** con Autentificación Keycloak **Coste medio** (aumenta la complejidad del software en caso de ser necesaria)
- **Coste por solapamiento**. **Coste alto**
- **Coste de operación Coste medio**

Alternativa 3

En esta alternativa planteamos el uso de Keycloak, concretamente la creación de clientes con autentificación basada en clientId y secret. En esta alternativa el productor solicita un token mediante el uso del client Id y un secret y a partir de ese momento usa el token para la comunicación mediante el estándar oauth2.0.

Posibilidades

- **Autenticación y autorización:** Los ClientId - Secrets permiten autenticar y autorizar a un sistema tercero generando un token de acceso a los recursos del sistema que estén autorizados.
- **Autorización:** Los ClientId - Secrets permiten autorizar a un sistema tercero a acceder a los recursos del sistema.
- **Flexibilidad:** Los ClientId - Secrets pueden ser utilizadas para restringir el acceso a los recursos del sistema en función de las necesidades de cada sistema tercero.
- **Seguridad:** Los ClientId - Secrets pueden ser utilizadas para proteger los recursos del sistema de accesos no autorizados.
- **Revocación:** El uso de clientes y tokens oauth garantizan la posibilidad de revocación.
- **Estándares de seguridad:** Keycloak se integra con Oauth1.0, Oauth2.0, SAML.

Ventajas

- **Fácil implementación:** Los clientes Keycloak son fáciles de implementar y utilizar y mantener. No requiere de herramientas adicionales
- **Escalabilidad:** Los clientes Keycloak son escalables y pueden ser utilizadas para gestionar un gran número de sistemas terceros.
- **Coste:** Los clientes Keycloak son una solución de bajo coste.
- **Flexibilidad:** Permite flujos complejos de autentificación y múltiples sistemas de autentificación.
- **Grano fino de autorización**

Desventajas

- **Requiere configuración:** Los clientes requieren configuración en Keycloak en el Realm correspondiente.

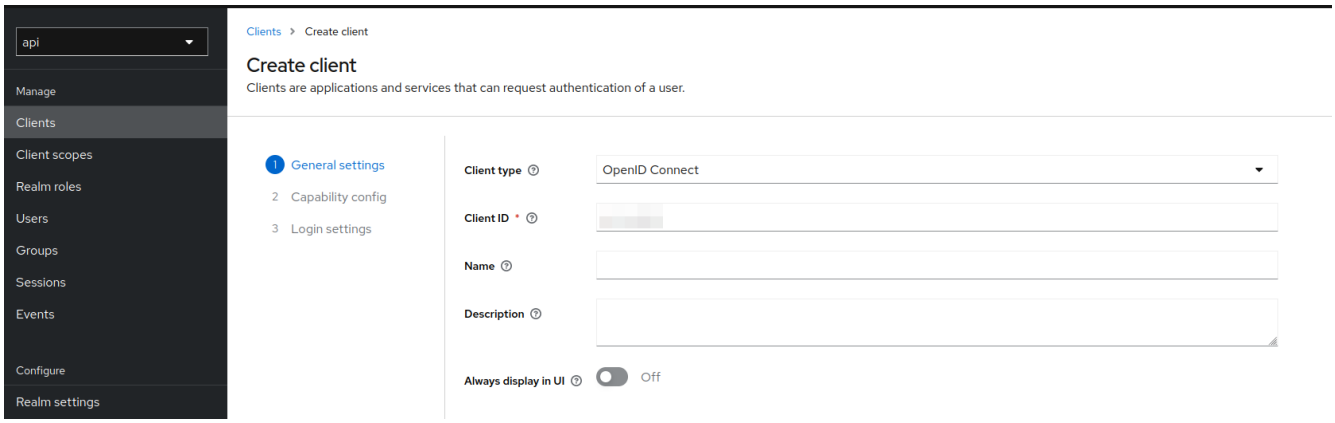
Coste Adopción

- **Coste sistema No aplica** (usa keycloak)
- **Coste Integración** con Autentificación Keycloak **No aplica** (Es el mismo sistema)
- **Coste por solapamiento**. **No aplica** (no existen duplicidades ya que solo un sistema)
- **Coste de operación Coste medio** (coste de configuración del cliente correspondiente)

Planteamos una pequeña POC de ejemplo

POC Client Id

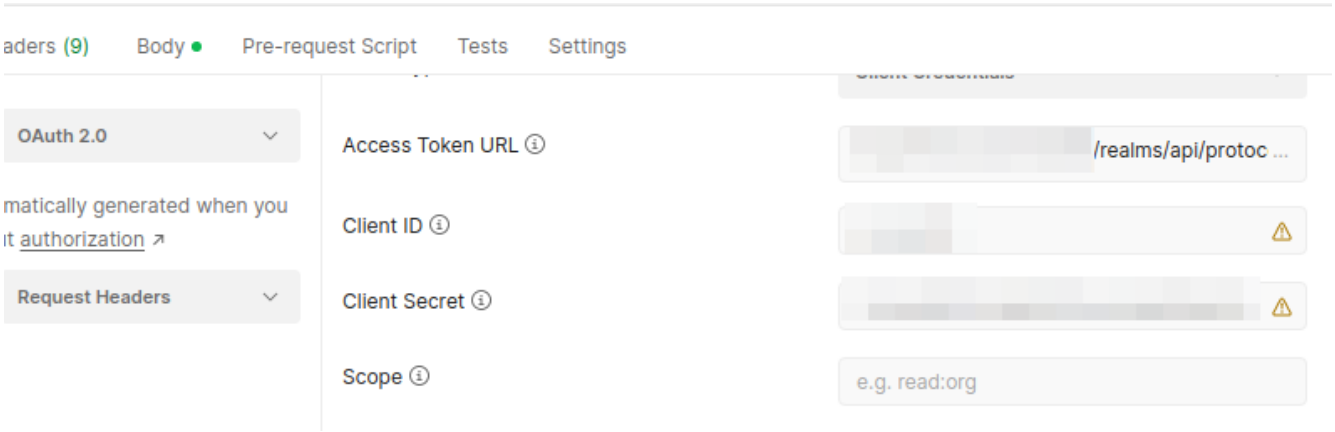
Planteamos un realm "api". Sobre este realm creamos un cliente "api-n" de tipo OpenID Connect tal y como se muestra en la captura siguiente.



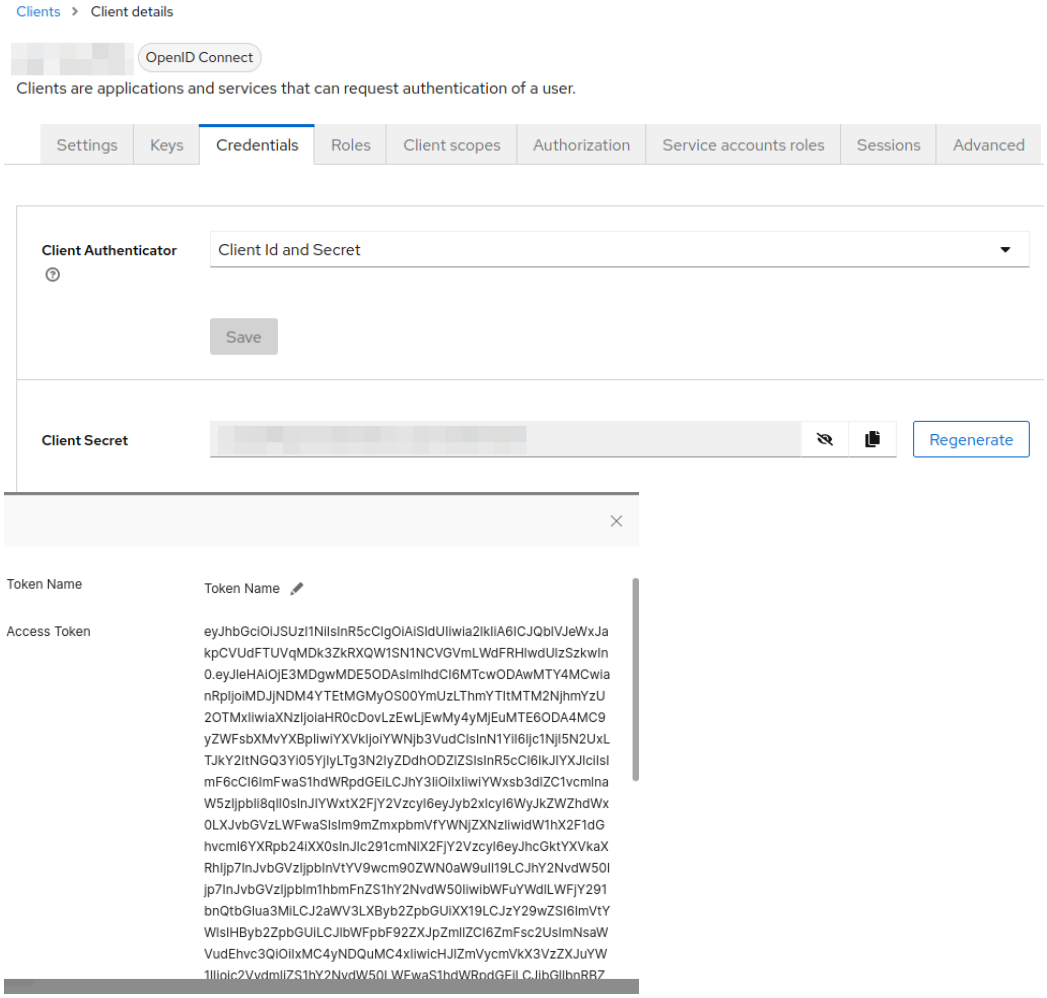
SAS

Dentro de las configuraciones de las capacidades de este cliente activamos "Client authentication" si deseamos una autorización de grano fino activamos la autorización. a continuación marcamos los flujos de autentificación que deseamos tener activos dentro de los parámetros OAuth2.0.

Una vez que hemos creado nuestro cliente acudimos a la pestaña Credentials, generamos un secret aleatorio e indicamos que la autentificación será mediante "Client Id and Secret". para una POC no entraremos en más detalles.



Cuando hacemos una petición postman para obtener el token podemos ver que la autentificación es correcta.



Podemos ver que el contenido del token es correcto y está activo, por lo tanto la autentificación ha sido correcta.

El impacto en el microservicio que debe validar el token hay que considerar los siguientes cambios:

microprofile-config.properties

```
mp.jwt.token.header=Authorization
mp.jwt.token.cookie=Bearer
```

server.xml

```
<feature>microProfile-3.3</feature>


<mpJwt id="defaultmpjwt" jwksUri="https://keycloak.ejemplo.org/auth/realms/sistema-n/protocol/openid-connect
/certs" issuer="https://keycloak.ejemplo.org/auth/realms/sistema-n"
        userNameAttribute="azp">
                <audiences>ALL_AUDIENCES</audiences>
</mpJwt>
```

El método que debe validar el token se debe anotar con: @RolesAllowed("LISTA-ROLES-AUTORIZADOS")

Decisión Tomada

Describe brevemente la decisión arquitectónica tomada.

Consideramos que la alternativa 3 es la que presenta un mejor balance coste beneficio, y que nos mantiene un modelo claro y sencillo, con capacidades de administración de los clientes autorizados a verter auditoria, manteniéndose además dentro de los estándares, y con un impacto en costes muy reducido tanto evidentes en MVP como ocultos cuando se inicie el proceso de integración en las aplicaciones terceras.

Keycloak ya proporciona una solución para el problema planteado esta solución se apoya en un estándar como OAuth2.0. Al usar la misma herramienta y los mismos recursos que para el resto de autentificaciones **evitamos introducir un sistema adicional, duplicidades, el aumento de la complejidad y solapamientos con los sobrecostes que conlleva**. mantenemos el sistema simple

El uso de un clientId con un secret asociado es en gran medida equivalente a la solución de un API-KEY dónde usamos un ID del cliente y una secret para obtener un token autentificado seguro con los roles y autorizaciones que le corresponda. Desde el punto de vista de la máquina invocante varia ligeramente el flujo ya que invoca a la URL proporcionada con el clientId y el password y obtiene el token generado por keycloak, a partir de este punto el funcionamiento/flujo es el estándar para cualquier autentificación por Oauth2.0

Al seguir la estrategia **con Keycloack no es necesario tener diferentes estrategias de seguridad. por ejemplo un servicio invocado por un usuario y por una máquina no necesita una autentificación OAUTH y apy-key, le basta con OAUTH.**

Keycloak es altamente configurable y adaptable permitiéndonos adaptar la seguridad al nivel exacto que se requiera para cada cliente; es mas completo y flexible. **Una vez establecida la configuración deseada se puede generar un JSON con dicha configuración e importarlo para los nuevos clientes, convirtiendo la creación de nuevos clientes en algo trivial.**

Al mantenernos en la opción de keycloak obtenemos una solución cercana a API-KEY en sencillez, con facilidad para operarlo e integrarlo en DEVOPS.

Consecuencias

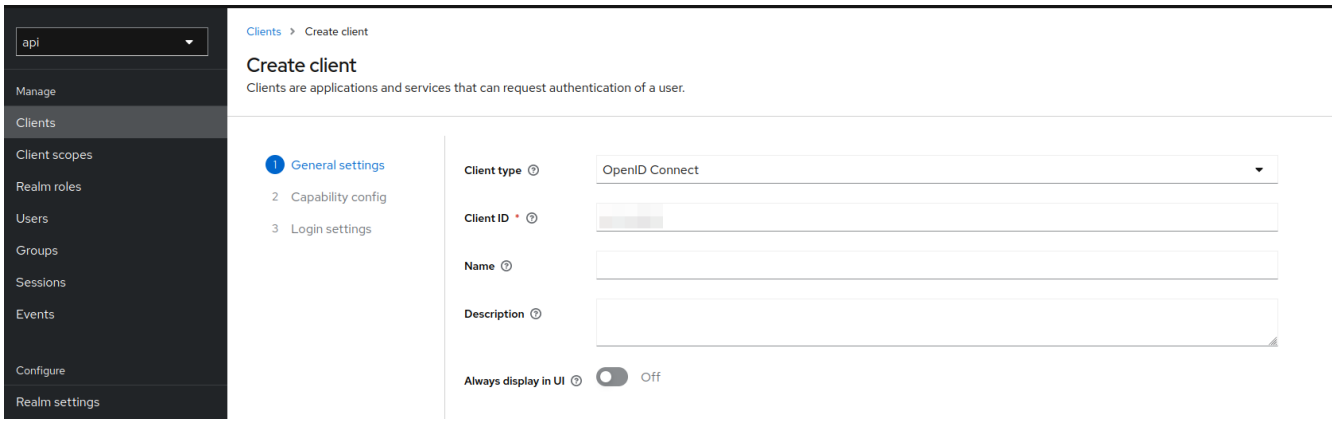
Describe brevemente la decisión arquitectónica tomada.

- Es necesario categorizar los clientes determinando las características que se requieren para cada tipo. Definidos en Keycloak los tipos de clientes y las configuraciones óptimas se puede mecanizar la creación de nuevos clientes.
- Con el uso de Keycloak se podrá disponer de un control claro y completo de todas las integraciones que se realizan con un API a través de la estructura REALM/CLIENT pudiendo revocar una integración concreta cuando sea necesaria.
- No se requerirá de nuevas herramientas en el esquema de seguridad ni aplicativos con esquemas de seguridad externos o independientes.
- Cualquier necesidad no cubierta se puede extender mediante SPI por lo que se deberá plantear y estudiar cada caso.
- El coste de adopción de esta solución en auditoria implica configurar tres ficheros con configuraciones muy concretas y fáciles de aplantillar por lo que se puede concluir que es un coste muy bajo.

Operación

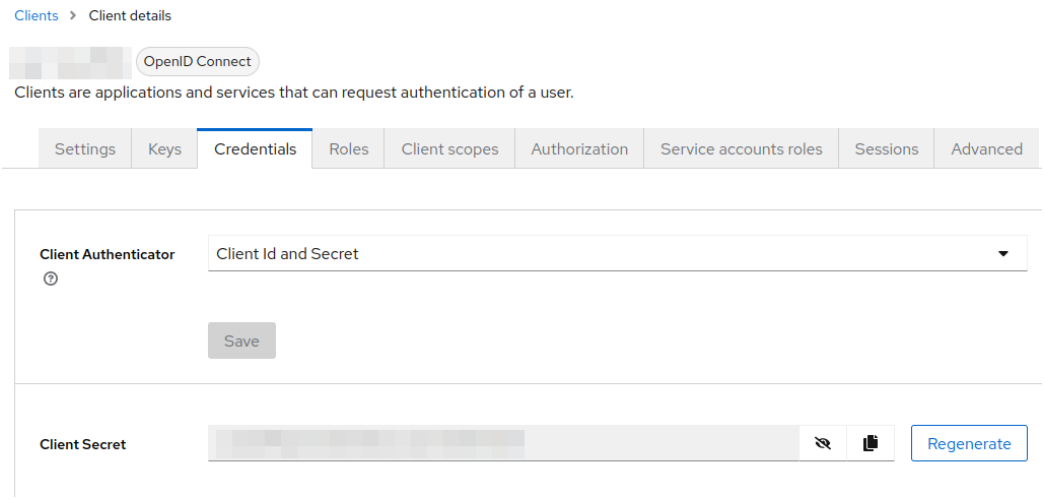
Creación Cliente

Creamos un cliente de tipo OpenID Connect en el realm deseado tal y como se muestra en la captura siguiente.



Dentro de las configuraciones de las capacidades de este cliente activamos "Client authentication" si deseamos una autorización de grano fino activamos la autorización. a continuación marcamos los flujos de autentificación que deseamos tener activos dentro de los parámetros OAuth2.0.

Una vez que hemos creado nuestro cliente acudimos a la pestaña Credentials, generamos un secret aleatorio e indicamos que la autentificación será mediante "Client Id and Secret". para una POC no entraremos en más detalles.



En los mapeos del token se requiere añadir upn y groups con los roles, ningún otro mapeo en especial.

Revocación:

Desde la consola de keycloak se puede revocar una sesión concreta de un cliente dado.

Deshabilitar el cliente o incluso eliminándolo causa que un clientId ya no pueda autentificar, lo equivale a una revocación aunque las sesiones activas continuarían vivas y una vez cerradas no se pueden crear nuevas para ese cliente.