**GRUPO 8**

**EXPERIMENTO 2 – ENTREGA FINAL – SEGURIDAD**

1. Autenticación y autorización para escribir en los tópicos del servidor de mensajería de Mosquitto

* 1. Mecanismo para definir los permisos que tiene cada rol sobre los topics:

Con el fin de facilitar las búsquedas y considerando el contexto IoT de la aplicación, se decidió utilizar una estructura Hash (soportada por Redis) para almacenar los permisos asociados con cada tópico. De esta forma, por cada uno, se tienen parejas llave-valor donde las llaves corresponden a los roles y los valores corresponden a los permisos que existen sobre el tópico. Por practicidad, los permisos cuentan con un identificador numérico definido de la siguiente forma:

|  |  |
| --- | --- |
| Identificador | Permiso |
| 1 | Lectura |
| 2 | Escritura |
| 3 | Lectura-Escritura |

Por consiguiente, cada tópico tiene asociada una estructura Hash, así:

|  |  |
| --- | --- |
| TOPICO i | |
| ‘rol1’ | #permiso |
| ‘rol2’ | #permiso |
| ‘rol3’ | #permiso |
| … | … |
| ‘rol n’ | #permiso |

Donde #permiso = 1, 2 ó 3

Se escogió esta implementación porque beneficia la inserción y borrado de tópicos, que en el contexto de la aplicación tiene una frecuencia mucho más alta que la de los posibles roles (considerando la cantidad de áreas, niveles y microcontroladores). Por otra parte, para realizar la verificación simplemente se obtiene la tabla Hash correspondiente al tópico y de allí se obtiene el valor asociado al rol del usuario que hizo la petición (que indica el permiso que tiene sobre el tópico). Lo anterior tiene complejidad temporal constante y por ende favorece el desempeño.

* 1. Mecanismo para asociar cada usuario a su rol:

Con el fin de establecer esta asociación, por cada usuario se tienen dos parejas llave-valor donde se almacena la contraseña y el rol. Es decir, se cuenta con la siguiente estructura:

|  |  |
| --- | --- |
| USUARIO i | |
| ‘contrasenia’ | ValorContraseña |
| ‘rol’ | RolAsociado |

Lo anterior complementa la implementación descrita previamente, puesto que en el momento de la verificación se obtiene fácilmente el rol correspondiente al usuario que realiza la petición. En particular, se busca por el nombre de usuario y de allí se solicita el valor asociado al ‘rol’. Una vez se obtiene dicho rol, este se utiliza para obtener los permisos que existen sobre el tópico en cuestión por medio de la tabla Hash mencionada en la sección anterior.

Se escogió esta implementación porque encapsula toda la información relevante del usuario y además facilita las búsquedas por nombre de usuario (que se realizan con frecuencia por cada petición de lectura o escritura), beneficiando el desempeño.

**Nota:** La implementación de las estrategias mencionadas en el presente documento se incluye en el archivo ‘permisos.py’ donde se observa la forma en que se persisten estas estructuras haciendo uso de las instrucciones hmset y hset asociadas al tipo Hash de Redis. Por su parte, los servicios REST de autenticación y autorización se encuentran en el archivo seguridadEntFisica.py

1. Descripción de los escenarios de experimentación:

Para realizar las pruebas de carga correspondientes, se creó un servicio REST para simular el ingreso de datos por parte de los microcontroladores asociados al sistema de minas. De esta forma, fue posible utilizar la herramienta JMeter, que se configuró con los siguientes parámetros:

* Number of Threads (microcontroladores): En ambos casos (con y sin seguridad) se incrementó este número de 500 en 500 y luego de 1000 en 1000 hasta que se degradó el desempeño.
* Ramp-Up Period: Para todas las pruebas se tiene un periodo de subida de 1 segundo, de forma tal que al incrementar el número de microcontroladores se pueda ver que tan bien se comporta el sistema si se envían más peticiones en la misma ventana de tiempo.
* Loop Count: Se estableció en 1 para todas las pruebas.

**Nota: lo anterior se observa con detalle en la tabla de resultados.**

## Resultados obtenidos y gráficas

Por cada prueba se registraron los valores obtenidos que dan cuenta del comportamiento del sistema según se incrementaba el número de usuarios. En particular, referente al desempeño se registraron los valores de latencia media, mínima y máxima, además del porcentaje de error y el rendimiento asociado. En ambos casos (con y sin seguridad) se ejecutaron pruebas hasta que el porcentaje de error en la escritura fuese superior a 0%. Asimismo, se indicó en qué momento el tiempo de respuesta supero los 30 ms para especificar en qué condiciones se degradó la latencia.

En particular, se observó que en casi todos los casos los tiempos de respuesta asociados a la implementación de las tácticas de seguridad son superiores a los de la aplicación sin estas tácticas. No obstante, se satisfacen las condiciones del escenario crítico (manejar notificaciones provenientes de 2500 microcontroladores con tiempos de respuesta inferiores a 30 ms). Del mismo modo, se ve que para el caso de la aplicación segura no fue posible escalar a 5000 microcontroladores puesto que la latencia media se degrada al punto de superar los 30 ms de respuesta (34 ms en específico).

**Nota: Se puede observar las gráficas en el archivo ReporteFinalExp2.xlsx**

1. Conclusiones

De lo anterior se observa que el desempeño se ve perjudicado cuando se implementan tácticas de seguridad, pero con las decisiones de diseño correctas es posible seguir satisfaciendo los escenarios propuestos. En particular, el uso de estructuras con acceso constante para almacenar la información asociada a la seguridad y el aprovechamiento de herramientas expuestas por proveedores son elementos clave para que se logre un buen balance entre la seguridad y el desempeño requerido por el sistema de minas.