Documentación Servidor Central

Generated by Doxygen 1.9.8

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 psk_valid_keys_t Struct Reference

Estructura para almacenar las credenciales válidas.

Public Attributes

- char ** keys
- int count
- int capacity

3.1.1 Detailed Description

Estructura para almacenar las credenciales válidas.

Esta estructura tiene todas las credenciales válidas cargadas desde el archivo de configuración. Utiliza un array dinámico para almacenar las credenciales como strings.

3.1.2 Member Data Documentation

3.1.2.1 capacity

```
int psk_valid_keys_t::capacity
```

Capacidad total del array

3.1.2.2 count

```
int psk_valid_keys_t::count
```

Número actual de credenciales cargadas

3.1.2.3 keys

```
char** psk_valid_keys_t::keys
```

Array de punteros a credenciales

The documentation for this struct was generated from the following file:

- src/psk_validator.c

Chapter 4

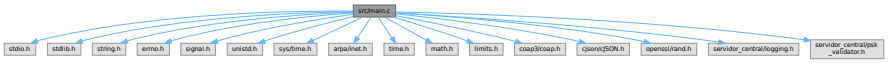
File Documentation

4.1 src/main.c File Reference

Servidor central de control de ascensores con protocolo CoAP sobre DTLS.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <signal.h>
#include <unistd.h>
#include <sys/time.h>
#include <arpa/inet.h>
#include <time.h>
#include <math.h>
#include <limits.h>
#include <coap3/coap.h>
#include <cjson/cJSON.h>
#include <openssl/rand.h>
#include "servidor_central/logging.h"
#include "servidor_central/psk_validator.h"
```

Include dependency graph for main.c:



Macros

- #define **PSK_SERVER_HINT** "ElevatorCentralServer"
- #define RESOURCE_FLOOR_CALL "peticion_piso"

  Ruta del recurso CoAP para peticiones de piso.
- #define RESOURCE_CABIN_REQUEST "peticion_cabina"

  Ruta del recurso CoAP para peticiones de cabina.
- #define SERVER_IP "0.0.0.0"

  Dirección IP de escucha del servidor.
- #define SERVER_PORT "5684"

  Puerto de escucha del servidor CoAP DTLS.

Functions

- void handle_sigint (int signum)

  Manejador de señal para SIGINT (Ctrl+C)
- void generate_unique_task_id (char *task_id_out, size_t len)

  Genera un ID único para una tarea de ascensor.
- int main (int argc, char **argv)

  Función principal del Servidor Central de Ascensores.

4.1.1 Detailed Description

Servidor central de control de ascensores con protocolo CoAP sobre DTLS.

Author

Sistema de Ascensores

Date

2024

Version

2.0

Este archivo implementa el servidor central que gestiona las solicitudes de ascensores desde múltiples API Gateways usando CoAP sobre DTLS con autenticación segura.

Arquitectura del sistema:

- Servidor CoAP-DTLS que escucha en puerto 5684
- Autenticación mutua mediante certificados seguros
- Procesamiento stateless de solicitudes de ascensores
- Balanceamiento de carga automático entre ascensores
- Generación temporal de IDs únicos
- Logging detallado para debugging y auditoría

Recursos CoAP disponibles:

- `POST /peticion_piso`: Solicitudes de llamada desde pisos
- `POST /peticion_cabina`: Solicitudes desde cabinas de ascensores

Seguridad:

- Cifrado DTLS para todas las comunicaciones
- Autenticación mutua con validación de identidades

- Timeouts configurables para estabilidad de conexiones
- Logging de todos los eventos de seguridad

Funcionamiento:

1. Inicialización del contexto CoAP con DTLS
2. Configuración de callbacks de autenticación
3. Registro de recursos CoAP disponibles
4. Bucle principal procesando solicitudes
5. Terminación con liberación de recursos

Note

Requiere libcoap compilado con soporte DTLS/OpenSSL

See also

https://libcoap.net/doc/reference/4.3.0/

4.1.2 Macro Definition Documentation

4.1.2.1 RESOURCE_CABIN_REQUEST

```
#define RESOURCE_CABIN_REQUEST "peticion_cabina"
```

Ruta del recurso CoAP para peticiones de cabina.

Define la ruta del endpoint CoAP que maneja las solicitudes de cabina (cabin requests) desde el interior de los ascensores.

4.1.2.2 RESOURCE_FLOOR_CALL

```
#define RESOURCE_FLOOR_CALL "peticion_piso"
```

Ruta del recurso CoAP para peticiones de piso.

Define la ruta del endpoint CoAP que maneja las solicitudes de llamada de piso (floor calls) desde botones externos.

4.1.2.3 SERVER_IP

```
#define SERVER_IP "0.0.0.0"
```

Dirección IP de escucha del servidor.

Dirección IP en la que el servidor CoAP escuchará conexiones. "0.0.0.0" indica que escuchará en todas las interfaces disponibles.

4.1.2.4 SERVER_PORT

```
#define SERVER_PORT "5684"
```

Puerto de escucha del servidor CoAP DTLS.

Puerto en el que el servidor CoAP con DTLS escuchará conexiones. El puerto 5684 es el puerto estándar para CoAP sobre DTLS.

4.1.3 Function Documentation

4.1.3.1 generate_unique_task_id()

```
void generate_unique_task_id (
            char * task_id_out,
            size_t len )
```

Genera un ID único para una tarea de ascensor.

Parameters

out	*task_id_out*	Buffer donde se almacenará el ID generado
in	*len*	Tamaño del buffer de salida en bytes

Esta función genera un identificador único para tareas de ascensor basado en el timestamp actual del sistema con precisión de milisegundos.

Formato del ID generado:
`"T_{segundos_unix}{milisegundos}"`

Ejemplo de ID:

- "T_1640995200123" donde:
  - T_: Prefijo identificador de tarea
  - 1640995200: Segundos desde epoch Unix
  - 123: Milisegundos (3 dígitos)

Características:

- Unicidad temporal garantizada
- Formato legible y ordenable
- Precisión de milisegundos
- Compatible con sistemas distribuidos

Limitaciones:

- No es thread-safe (para entornos multihilo usar sincronización)

- Dependiente del reloj del sistema
- Máximo 1,000 IDs por segundo (limitación de milisegundos)

Uso típico:

```
char task_id[32];
generate_unique_task_id(task_id, sizeof(task_id));
// task_id contiene "T_1640995200123"
```

Note

El buffer de salida debe tener al menos 32 caracteres

Para entornos multihilo considerar usar mutex o contadores atómicos

See also

gettimeofday()

snprintf()

4.1.3.2 handle_sigint()

```
void handle_sigint (
            int signum )
```

Manejador de señal para SIGINT (Ctrl+C)

Parameters

in	*signum*	Número de señal recibida (se espera SIGINT = 2)

Esta función implementa el manejo de la señal SIGINT:

Funcionalidad:

- Establece la bandera global 'running' a 0
- Permite que el bucle principal termine de forma controlada
- Registra el evento en el log del sistema
- Evita terminación abrupta del servidor

Flujo de terminación:

1. Usuario presiona Ctrl+C
2. Sistema envía SIGINT al proceso
3. Esta función establece running = 0
4. Bucle principal detecta el cambio y termina
5. Se ejecutan rutinas de limpieza en main()

Seguridad:

- No realiza operaciones complejas en el handler
- Solo modifica la bandera de control
- Es thread-safe para el contexto de señales

Note

Esta función debe ser registrada con signal() o sigaction()

Solo modifica variables globales para evitar problemas de reentrancia

See also

running

main()

4.1.3.3 main()

```
int main (
            int argc,
            char ** argv )
```

Función principal del Servidor Central de Ascensores.

Parameters

in	*argc*	Número de argumentos de línea de comandos
in	*argv*	Array de argumentos de línea de comandos

Returns

EXIT_SUCCESS (0) si el servidor termina correctamente, EXIT_FAILURE (1) en caso de error

Esta función implementa el punto de entrada principal del servidor central de control de ascensores. Configura y ejecuta un servidor CoAP con DTLS-PSK que gestiona solicitudes de ascensores desde API Gateways.

Flujo de inicialización:

1. **Configuración de señales**: Registra manejador para SIGINT (Ctrl+C)
2. **Inicialización de libCoAP**: Configura logging y contexto CoAP
3. **Configuración de red**: Establece dirección y puerto de escucha
4. **Configuración DTLS-PSK**: Configura autenticación y cifrado
5. **Inicialización PSK**: Carga validador de claves precompartidas
6. **Registro de recursos**: Configura endpoints CoAP

7. **Bucle principal**: Procesa solicitudes hasta terminación

Configuración de seguridad:

- Autenticación DTLS-PSK con claves precompartidas
- Validación de identidades de clientes
- Cifrado de todas las comunicaciones
- Timeouts optimizados para estabilidad

Recursos CoAP registrados:

- `POST /peticion_piso`: Solicitudes de llamada de piso
- `POST /peticion_cabina`: Solicitudes de cabina

Gestión de errores:

- Validación de configuración de red
- Verificación de inicialización de componentes
- Logging detallado de errores críticos
- Terminación en caso de fallo

Terminación:

- Respuesta a señal SIGINT (Ctrl+C)
- Cierre de conexiones DTLS
- Limpieza de contexto CoAP

Configuración de red:

- Puerto: 5684 (estándar CoAP-DTLS)
- Interfaz: 0.0.0.0 (todas las interfaces)
- Protocolo: UDP con DTLS

Note

El servidor se ejecuta indefinidamente hasta recibir SIGINT
Requiere archivo de claves PSK para funcionamiento completo

See also

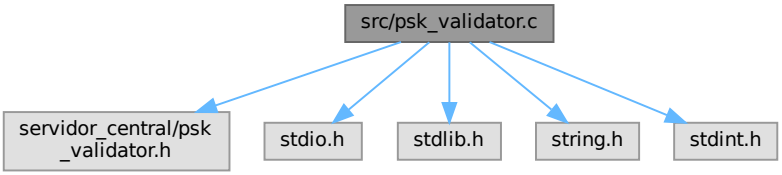
handle_sigint()
session_event_handler()
get_psk_info()
psk_validator_init()
hnd_floor_call()
hnd_cabin_request()

4.2 src/psk_validator.c File Reference

Implementación del validador de credenciales para autenticación DTLS.

```
#include "servidor_central/psk_validator.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
```

Include dependency graph for psk_validator.c:



Classes

- struct psk_valid_keys_t

  Estructura para almacenar las credenciales válidas.

Functions

- int psk_validator_init (const char ∗keys_file_path)

  Inicializa el validador de credenciales.
- int psk_validator_check_key (const char ∗key, size_t key_len)

  Valida si unas credenciales están en la lista de credenciales válidas.
- int psk_validator_get_key_for_identity (const char ∗identity, uint8_t ∗key_buffer, size_t buffer_size)

  Obtiene credenciales válidas para una identidad específica.
- int psk_validator_get_key_by_index (int index, uint8_t ∗key_buffer, size_t buffer_size)

  Obtiene credenciales por índice específico.
- void psk_validator_cleanup (void)

  Libera los recursos del validador de credenciales.
- int psk_validator_get_key_count (void)

  Obtiene el número total de credenciales disponibles.
- int psk_validator_is_initialized (void)

  Verifica si el validador está inicializado.

4.2.1 Detailed Description

Implementación del validador de credenciales para autenticación DTLS.

Author

Sistema de Control de Ascensores

Version

2.0

Date

2025

Este archivo implementa el sistema de validación de credenciales utilizado en la autenticación DTLS del servidor central. El sistema carga credenciales desde un archivo de configuración y proporciona funciones para validar y obtener credenciales de forma determinística basada en la identidad del cliente.

See also

psk_validator.h

dtls_common_config.h

4.2.2 Function Documentation

4.2.2.1 psk_validator_check_key()

```
int psk_validator_check_key (
            const char * key,
            size_t key_len )
```

Valida si unas credenciales están en la lista de credenciales válidas.

Parameters

in	*key*	Credenciales a validar
in	*key_len*	Longitud de las credenciales en bytes

Returns

1 si las credenciales son válidas, 0 si no lo son

Esta función valida unas credenciales contra la lista de credenciales válidas:

- Verifica que haya credenciales cargadas en memoria
- Compara las credenciales proporcionadas con cada credencial almacenada

- Utiliza comparación exacta de longitud y contenido
- Retorna el resultado de la validación

Note

La búsqueda es lineal O(n) donde n es el número de credenciales
La función es thread-safe para lecturas concurrentes

See also

psk_validator_get_key_for_identity

4.2.2.2 psk_validator_cleanup()

```
void psk_validator_cleanup (
            void  )
```

Libera los recursos del validador de credenciales.

Esta función limpia todos los recursos asociados al validador:

- Libera cada credencial individual del array
- Libera el array de punteros a credenciales
- Resetea los contadores y punteros
- Prepara el validador para una nueva inicialización

Note

Debe ser llamada al finalizar para evitar memory leaks
Es seguro llamar esta función múltiples veces

See also

psk_validator_init

4.2.2.3 psk_validator_get_key_by_index()

```
int psk_validator_get_key_by_index (
            int index,
            uint8_t * key_buffer,
            size_t buffer_size )
```

Obtiene credenciales por índice específico.

Parameters

in	*index*	Índice de las credenciales en el archivo (0-based)
out	*key_buffer*	Buffer donde se almacenarán las credenciales
in	*buffer_size*	Tamaño del buffer en bytes

Returns

0 si se obtuvo correctamente, -1 en caso de error

Esta función obtiene credenciales por su posición en el array:

- Valida que el índice esté dentro del rango válido
- Obtiene las credenciales del array de credenciales cargadas
- Copia las credenciales al buffer proporcionado
- Verifica que el buffer tenga espacio suficiente

Note

El índice debe estar entre 0 y count-1

See also

psk_validator_get_key_for_identity

4.2.2.4 psk_validator_get_key_count()

```
int psk_validator_get_key_count (
            void  )
```

Obtiene el número total de credenciales disponibles.

Returns

Número de credenciales en el archivo

Esta función retorna el número total de credenciales que han sido cargadas desde el archivo de configuración.

Note

Solo es válida después de llamar a psk_validator_init

4.2.2.5 psk_validator_get_key_for_identity()

```
int psk_validator_get_key_for_identity (
            const char * identity,
            uint8_t * key_buffer,
            size_t buffer_size )
```

Obtiene credenciales válidas para una identidad específica.

Parameters

in	*identity*	Identidad del cliente
out	*key_buffer*	Buffer donde se almacenarán las credenciales
in	*buffer_size*	Tamaño del buffer en bytes

Returns

0 si se obtuvo correctamente, -1 en caso de error

Esta función obtiene las credenciales correspondientes a una identidad:

- Calcula un hash determinístico de la identidad
- Usa el hash como índice para seleccionar unas credenciales
- Copia las credenciales seleccionadas al buffer proporcionado
- Verifica que el buffer tenga espacio suficiente

Note

La selección es determinística: misma identidad = mismas credenciales

El algoritmo usa hash simple para distribución uniforme

See also

psk_validator_check_key

4.2.2.6 psk_validator_init()

```
int psk_validator_init (
            const char * keys_file_path )
```

Inicializa el validador de credenciales.

Parameters

in	*keys_file_path*	Ruta al archivo de configuración de autenticación

Returns

0 si se inicializó correctamente, -1 en caso de error

Esta función inicializa el sistema de validación de credenciales:

- Abre el archivo de configuración especificado
- Cuenta el número de líneas en el archivo

- Asigna memoria dinámica para almacenar las credenciales
- Lee todas las credenciales del archivo y las almacena en memoria
- Cierra el archivo después de la carga

Note

El archivo debe contener una credencial por línea

La función maneja automáticamente la memoria dinámica

See also

psk_validator_cleanup

4.2.2.7 psk_validator_is_initialized()

```
int psk_validator_is_initialized (
            void  )
```

Verifica si el validador está inicializado.

Returns

1 si está inicializado, 0 en caso contrario

Esta función verifica si el validador de credenciales ha sido inicializado correctamente y está listo para su uso.

Note

Útil para verificar el estado antes de usar otras funciones

Index