PRÁCTICA 3 – COMUNICACIÓN SEGURA CON SSL

Mario Valdemaro García Roqué y Roberto García Teodoro – Grupo 2313 – Pareja 7

Advertencias importantes.

1. Hemos realizado la creación de los certificados escribiéndolos en una carpeta oculta llamada .cert en el directorio home. Esto es debido a que necesitamos saber dónde se encuentran los certificados para llamar correctamente a las funciones de ssl.
2. Tal vez haya que repetir la ejecución un par de veces, ya que a veces encontramos errores que no nos dio tiempo a solventar.
3. Hay veces que si el cliente recibe ciertos mensajes del servidor SSL da una violación de segmento.

Server eco:

Hemos realizado un programa que hace las funciones de servidor-cliente para demostrar que se mandaban los mensajes de forma correcta utilizando ssl, para ello seguimos los siguientes pasos:

En primer lugar implementamos las nuevas funciones ssl descritas en el enunciado.

Después procedimos a implementar un cliente y un servidor.

Funciones implementadas:

*SSL_CTX** **inicializar_nivel_SSL**(int * sock);*

Argumentos:

- **Sock:**
  Inicializa y abre un socket que nos pasan por argumento, que no tiene por qué estar reservado.

Retorno:

Devuelve la estructura ctx donde se insertaran los certificados.

int **fijar_contexto_SSL**(SSL_CTX* ctx, const char* CAfile, const char* prvKeyFile,const char* certFile);*

Argumentos:

- **ctx:**
  Estructura ctx donde se cargaran los certificados.
- **CAfile:**
  Ruta hasta el fichero que contiene el certificado de la entidad certificadora.
- **prvKeyFile**
  Ruta hasta el fichero que contiene la clave privada.
- **certFile:**
  Ruta hasta el fichero que contiene el certificado.

Retorno:

Devuelve 1 en caso de éxito y 0 en caso de error.

*SSL** **conectar_canal_seguro_SSL**(SSL_CTX* ctx,int sock,struct sockaddr res);*

Argumentos:

- **ctx:**
  Estructura que contiene la información con los certificados.
- **sock:**
  Socket mediante el que realizaremos la conexión.
- **res:**
  Estructura con la información de la dirección a la que nos vamos a conectar.

Retorno:

Devuelve una estructura SSL con la información (certificados, etc.) del servidor al que nos hemos conectado.

*SSL** **aceptar_canal_seguro_SSL***(SSL_CTX* ctx,int sockfd,int puerto,int tam,struct sockaddr_in ip4addr);*

Argumentos:

- **ctx:**
  Estructura que contiene la información con los certificados.
- **sockfd:**
  Socket mediante el que realizaremos la conexión.
- **Puerto:**
  Puerto al que ligaremos el socket para recibir las peticiones de conexión.
- **Ip4addr:**
  Estructura que una vez ejecutada esta función contendrá la información de red del cliente que se nos ha conectado.

Retorno:

Devuelve una estructura SSL con la información (certificados, etc.) del cliente que se ha conectado con nosotros.

int **evaluar_post_connectar_SSL***(SSL * ssl);*

Argumentos:

- **ssl**:
  Estructura con la información de los certificados del cliente que se nos ha conectado.

Retorno:

Devuelve 1 si los certificados son válidos 0 sino.

int **enviar_datos_SSL***(SSL * ssl,const void * buf);*

Argumentos:

- **ssl:**
  Información del servidor/cliente/par al que estamos conectados.
- **buf:**
  Mensaje que deseamos enviar.

Retorno:

0 en caso de error, longitud del mensaje enviado en caso de éxito.

int **recibir_datos_SSL***(SSL * ssl, void * buf);*

Argumetos:

- **ssl:**
  Información del servidor/cliente/par al que estamos conectados.
- **buf:**
  Mensaje que hemos recibido.

Retorno:

0 en caso de error, longitud del mensaje recibido.

void **cerrar_canal_SSL***(SSL *ssl,SSL_CTX *ctx, int sockfd);*

Argumentos:

- **ssl:**
  Certificados del servidor/cliente/par al que estamos conectados.
- **ctx:**
  Estructura con los certificados que tenemos.
- **sockfd:**
  Socket que estamos usando

Retorno:

Nada

Estructura del servidor y el cliente eco:

```
void servidor(){
        int sockfd;
        SSL_CTX* ctx;
        SSL * ssl=NULL;
        char buf[8096]="hola mundo";
        struct sockaddr_in ip4addr;
        ctx = inicializar_nivel_SSL(&sockfd);
        ERR_print_errors_fp(stdout);
        fijar_contexto_SSL(ctx,"cert/root.pem",
"cert/server.pem","cert/server.pem");
        ERR_print_errors_fp(stdout);
        printf("aceptar\n");
        ssl=aceptar_canal_seguro_SSL(ctx,sockfd,
8080,80,ip4addr);
        ERR_print_errors_fp(stdout);
        printf("evaluar\n");
        if(!evaluar_post_connectar_SSL(ssl)){
                ERR_print_errors_fp(stdout);
                return;
        }
        printf("enviar\n");
        enviar_datos_SSL(ssl,buf);
        ERR_print_errors_fp(stdout);
        recibir_datos_SSL(ssl, buf);
        printf("recibido:[%s]\n",buf );
        cerrar_canal_SSL(ssl,ctx,sockfd);
}
```

```
void  cliente(){
        int sockfd;
        SSL_CTX* ctx=NULL;
        SSL * ssl=NULL;
        struct addrinfo hints, *res;
        char buf [20];
        memset(&hints, 0, sizeof(hints));
        hints.ai_family = AF_UNSPEC;
        hints.ai_socktype = SOCK_STREAM;
        printf("inicializar_nivel_SSL\n");
        ctx = inicializar_nivel_SSL(&sockfd);
        ERR_print_errors_fp(stdout);
        if(ctx==NULL){
                printf("CTX NULL\n");
                return;
        }
        printf("fijar_contexto_SSL\n");
        fijar_contexto_SSL(ctx,"cert/root.pem",
"cert/client.pem","cert/client.pem");
        ERR_print_errors_fp(stdout);
        if(0!=getaddrinfo("localhost", "8080", &hints, &res)){
                printf("Error al obtener informacion del
servidor\n");
                return;
        }
        printf("%p\n",(void*)res );
        printf("conexion\n");
        ssl=conectar_canal_seguro_SSL(ctx,sockfd,*(res-
>ai_addr));
        ERR_print_errors_fp(stdout);
        printf("evaluar\n");
        if(!evaluar_post_connectar_SSL(ssl)){
                ERR_print_errors_fp(stdout);
                return;
        }
        printf("recibir\n");
        recibir_datos_SSL(ssl, buf);
        printf("recibido:[%s]\n",buf );
```

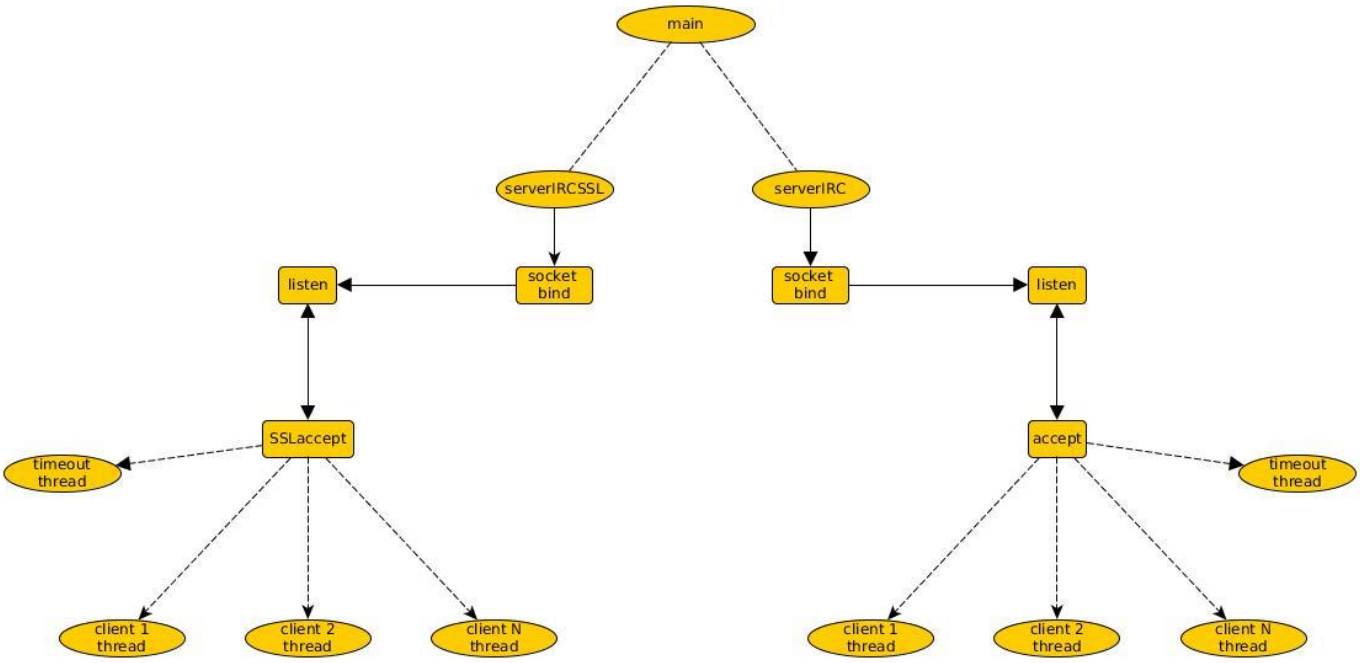
	enviar_datos_SSL(ssl, buf); *ERR_print_errors_fp(stdout);* *cerrar_canal_SSL(ssl,ctx,sockfd);* *freeaddrinfo(res);* *}*

El funcionamiento es sencillo, tras el handshake el servidor envía un Hola mundo, el cliente lo recibe y lo envía también, cerrando el canal después. El servidor recibe el Hola mundo del cliente y cierra el canal.

Para ejecutar, se aporta un main, se debe ejecutar de la forma ‘./G-2313-07-P3eco s’ para crear el servidor y posteriormente ‘./G-2313-07-P3eco c’ creará el cliente y comenzará la ejecución del programa.

Server irc:

La estructura del servidor es la siguiente:



Tenemos un hilo principal (main) que creara 2 procesos (serverIRCSSL, serverIRC), el motivo de crear 2 procesos es que ya que el servidor con SSL y el servidor normal no tienen que estar coordinados ni compartir recursos es mas sencillo crear 2 procesos independientes que 2 hilos que necesitaran de nuevas variables para que no se pisen el trabajo una parte y la otra, ya que son servidores separados.

Ambos servidores tienen una estructura idéntica y los cambios que se han llevado acabo son al iniciar la conexión, es decir aceptar un nuevo cliente, y al enviar y recibir mensajes. Para esto ultimo hemos adaptado la librería de conexión que teníamos para que en estos puntos usase las funciones de envío y recepción SSL en vez de las funciones de socket. Para que esto funcionase sin "depender" de librerías hemos añadido unas llamadas externas a unas funciones de las que depende.

```
extern void * WhoseIsThisSocket(int socket);
extern SSL * getSSL(void * client);
```

Aparte de eso no se han realizado mas cambios.

Cliente irc:

Los 2 únicos cambios que se han realizado en esta parte es a la hora de conectar con un servidor si detectamos que la conexión se va a realizar en el puerto 6697 entonces hacemos una conexión mediante SSL sino realizamos una conexión normal.

El otro cambio que se ha realizado es añadir una variable global

```
SSL * ssl;
```

que sera donde se guardara la información sobre el servidor para poder mandar y recibir la información, al mismo tiempo hemos introducido un cambio en la librería de conexión para que los mensajes se emitan por una conexión SSL si se requiere.

Los cambios realizados en la librería de conexión no se encuentran en la librería equivalente del servidor, al mismo tiempo los cambios realizados en el servidor sobre esta librería no se encuentran en la librería del cliente.

Multimedia:

No hemos podido realizar esta parte por falta de tiempo.