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:

Cliente irc:

1. Implementamos las nuevas funciones de la librería.
2. Cambiamos la función *void connect\_client(void)* para permitir que se conecte un cliente por ssl.

Multimedia:

No hemos podido realizar esta parte por falta de tiempo.