Krótkie wprowadzenie do korzystania z OpenSSL

Literatura:

http://www.openssl.org

E. Rescola, "An introduction to OpenSSL Programming (PartI)"
(http://www.linuxjournal.com/article/4822)
"An introduction to OpenSSL Programming (PartII)"
(http://www.linuxjournal.com/article/5487)
http://www.rtfm.com/openssl-examples/

Biblioteka OpenSSL

Pliki nagłówkowe

```
#include <openssl/ssl.h>
#include <openssl/error.h>
```

Kompilacja programu

```
gcc -o serwer serwer.c -lcrypto -lssl
gcc -o klient klient.c -lcrypto -lssl
```

Inicjalizacja biblioteki

```
#include <openssl/ssl.h>
#include <openssl/err.h>

SSL_library_init();
SSL_load_error_strings();
OpenSSL_add_all_algoritms();
```

Struktury wykorzystywane przez bibliotekę

SSL_METHOD

Pozwala określić metodę kryptograficzną wykorzystywana do komunikacji: SSLv1, SSLv2, ... , TLSv1

```
SSL_METHOD *my_ssl_method;
my_ssl_method = TLSv1_method();
```

SSL_CTX

Określa kontekst komunikacji serwera, czyli jakiej konfiguracji oczekujemy. Wykorzystywana jest do tworzenia obiektu reprezentującego każde połączenie.

```
SSL_CTX *my_ssl_ctx;

// utwórz nowy kontekst
my_ssl_ctx = SSL_CTX_new(my_ssl_method);

// lokalizacja plików z kluczem prywatnym i certyfikatów
SSL_CTX_use_certificate_file(my_ssl_ctx,"server.pem",SSL_FILETYPE_PEM);
SSL_CTX_usePrivateKey_file(my_ssl_ctx,"server.pem",SSL_FILETYPE_PEM);

//Weryfikacja klucza prywatnego
if (SSL(SSL_CTX_check_private_key(my_ssl_ctx)
// klucz działa
else
// niepoprawny klucz
```

BIO

Interfejs pozwalający czyta/zapisywać dane z różnych źródeł we/wy (gniazd, terminala, buforów pamięci, itd.)

SSL

Struktura zarządzająca danymi niezbędnymi do korzystania z bezpiecznego połączenia. Jest tworzona dla każdego połączenia.

```
SSL *my_ssl;
// połączenie struktury z kontekstem
my_ssl = SSL_new(my_ssl_ctx);

// połączenie struktury z gniazdem opisanym za pomocą deskryptora fd
SSL_set_fd(my_ssl, fd)

//server
if (SSL_accept(my_ssl) <= 0)
  // wystapiły błędy
else
  // nawiązano bezpieczne połączenie

//klient
if (SSL_connect(my_ssl) <= 0)
  // wystapiły błędy
else
  // nawiązano bezpieczne połączenie

// uzyskanie informacji o połączeniu
printf("[%s,%s]\n", SSL_get_version(my_ssl), SSL_get_cipher(my_ssl));
```

A. Przykład klienta

Działamy w oparciu o powiązanie z deskryptorami plików.

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <arpa/inet.h>
/*
    SSL includes
 */
#include <openssl/ssl.h>
#include <openssl/err.h>

int main(int argc, char *argv[]) {
    SSL_METHOD *my_ssl_method;
    SSL_CTX *my_ssl_ctx;
    SSL *my_ssl;

    int my_fd;
    struct sockaddr_in server;
    int error = 0, read_in = 0;
    char buffer[512];

    memset(buffer,'\0',sizeof(buffer));

    OpenSSL_add_all_algorithms();
    SSL_library_init();
    SSL_load_error_strings();

    my_ssl_method = TLSv1_client_method();

    if((my_ssl_ctx = SSL_CTX_new(my_ssl_method)) == NULL) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    if((my_ssl = SSL_new(my_ssl_ctx)) == NULL) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    my_fd = socket(AF_INET, SOCK_STREAM, 0);
    bzero(&server,sizeof(server));
    server.sin_family = AF_INET;
    server.sin_port = htons(5353);
    inet_aton("127.0.0.1",&server.sin_addr);
    bind(my_fd, (struct sockaddr *)&server, sizeof(server));
    connect(my_fd,(struct sockaddr *)&server, sizeof(server));


    SSL_set_fd(my_ssl,my_fd);
```

```
    if(SSL_connect(my_ssl) <= 0) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    printf("Connection made with [version,cipher]:
            [%s,%s]\n",SSL_get_version(my_ssl),SSL_get_cipher(my_ssl));

    for( read_in = 0; read_in < sizeof(buffer); read_in += error ) {
        error = SSL_read(my_ssl,buffer+read_in,sizeof(buffer) - read_in);
        if(error <= 0)
            break;
    }

    SSL_shutdown(my_ssl);
    SSL_free(my_ssl);
    SSL_CTX_free(my_ssl_ctx);
    close(my_fd);

    printf("%s",buffer);

    return 0;
}
```

A. Przykład serwera

Działamy w oparciu o powiązanie z deskryptorami plików.

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <arpa/inet.h>

/* SSL */
#include <openssl/ssl.h>
#include <openssl/err.h>

int main(int argc, char *argv[]) {
    SSL_METHOD *my_ssl_method;
    SSL_CTX *my_ssl_ctx;
    SSL *my_ssl;
    int my_fd,client_fd;
    struct sockaddr_in server, client;
    int client_size;
    int error = 0, wrote = 0;
    char buffer[] = "Hello there! Welcome to the SSL test server.\n\n";

    OpenSSL_add_all_algorithms();
    SSL_library_init();
    SSL_load_error_strings();

    my_ssl_method = TLSv1_server_method();

    if((my_ssl_ctx = SSL_CTX_new(my_ssl_method)) == NULL) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    SSL_CTX_use_certificate_file(my_ssl_ctx,"server.pem",SSL_FILETYPE_PEM);
    SSL_CTX_use_PrivateKey_file(my_ssl_ctx,"server.pem",SSL_FILETYPE_PEM);

    if(!SSL_CTX_check_private_key(my_ssl_ctx)) {
        fprintf(stderr,"Private key does not match certificate\n");
        exit(-1);
    }

    my_fd = socket(PF_INET, SOCK_STREAM, 0);
    server.sin_family = AF_INET;
    server.sin_port = htons(5353);
    server.sin_addr.s_addr = INADDR_ANY;
    bind(my_fd, (struct sockaddr *)&server, sizeof(server));
    listen(my_fd, 5);
```

```
    for(;;) {
        client_size = sizeof(client);
        bzero(&client,sizeof(client));
        client_fd = accept(my_fd, (struct sockaddr *)&client,
                    (socklen_t *)&client_size);
        if((my_ssl = SSL_new(my_ssl_ctx)) == NULL) {
            ERR_print_errors_fp(stderr);
            exit(-1);
        }

        SSL_set_fd(my_ssl,client_fd);

        if(SSL_accept(my_ssl) <= 0) {
            ERR_print_errors_fp(stderr);
            exit(-1);
        }

        printf("Connection made with [version,cipher]:
                [%s,%s]\n",SSL_get_version(my_ssl),SSL_get_cipher(my_ssl));

        for(wrote = 0; wrote < strlen(buffer); wrote += error) {
            error = SSL_write(my_ssl,buffer+wrote,strlen(buffer)-wrote);

            if(error <= 0)
                break;
        }

        SSL_shutdown(my_ssl);

        SSL_free(my_ssl);
        close(client_fd);
    }

    SSL_CTX_free(my_ssl_ctx);

    return 0;
}
```

B. Przykład klienta

Wykorzystywana jest struktura BIO

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <string.h>
/* SSL  */
#include <openssl/ssl.h>
#include <openssl/err.h>

int main(int argc, char *argv[]) {
    SSL_METHOD *my_ssl_method;
    SSL_CTX *my_ssl_ctx;
    SSL *my_ssl;
    BIO *my_bio;
    int error = 0, read_in = 0;
    char buffer[512];

    memset(buffer,'\0',sizeof(buffer));

    OpenSSL_add_all_algorithms();
    SSL_library_init();
    SSL_load_error_strings();

    my_ssl_method = TLSv1_client_method();

    if((my_ssl_ctx = SSL_CTX_new(my_ssl_method)) == NULL) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    if((my_ssl = SSL_new(my_ssl_ctx)) == NULL) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    if((my_bio = BIO_new_connect("127.0.0.1:5353")) == NULL) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    if(BIO_do_connect(my_bio) <=0) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    SSL_set_bio(my_ssl,my_bio,my_bio);

    if(SSL_connect(my_ssl) <= 0) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    printf("Connection made with [version,cipher]:
            [%s,%s]\n",SSL_get_version(my_ssl),SSL_get_cipher(my_ssl));
```

```
    for( read_in = 0; read_in < sizeof(buffer); read_in += error ) {
        error = SSL_read(my_ssl,buffer+read_in,sizeof(buffer) - read_in);
        if(error <= 0)
            break;
    }

    SSL_shutdown(my_ssl);
    SSL_free(my_ssl);
    SSL_CTX_free(my_ssl_ctx);

    printf("%s",buffer);

    return 0;

    report_error("Report error (not quit) test\n",__FILE__,__LINE__,0);
    report_error_q("Report error (quit) test\n",__FILE__,__LINE__,0);
    return 0;
}
```

B. Przykład serwera

Wykorzystywana jest struktura BIO

```
/*
    Standard includes
 */
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <string.h>
/*
    SSL includes
 */
#include <openssl/ssl.h>
#include <openssl/err.h>

int main(int argc, char *argv[]) {
    SSL_METHOD *my_ssl_method;
    SSL_CTX *my_ssl_ctx;
    SSL *my_ssl;
    BIO *server_bio,*client_bio;
    int error = 0, wrote = 0;
    char buffer[] = "Hello there! Welcome to the SSL test server.\n\n";

    OpenSSL_add_all_algorithms();
    SSL_library_init();
    SSL_load_error_strings();

    my_ssl_method = TLSv1_server_method();

    if((my_ssl_ctx = SSL_CTX_new(my_ssl_method)) == NULL) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    SSL_CTX_use_certificate_file(my_ssl_ctx,"server.pem",SSL_FILETYPE_PEM);
    SSL_CTX_use_PrivateKey_file(my_ssl_ctx,"server.pem",SSL_FILETYPE_PEM);

    if(!SSL_CTX_check_private_key(my_ssl_ctx)) {
        fprintf(stderr,"Private key does not match certificate\n");
        exit(-1);
    }

    if((server_bio = BIO_new_accept("5353")) == NULL) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }

    if(BIO_do_accept(server_bio) <= 0) {
        ERR_print_errors_fp(stderr);
        exit(-1);
    }
```

```
    for(;;) {
        if(BIO_do_accept(server_bio) <= 0) {
            ERR_print_errors_fp(stderr);
            exit(-1);
        }

        client_bio = BIO_pop(server_bio);

        if((my_ssl = SSL_new(my_ssl_ctx)) == NULL) {
            ERR_print_errors_fp(stderr);
            exit(-1);
        }

        SSL_set_bio(my_ssl,client_bio,client_bio);

        if(SSL_accept(my_ssl) <= 0) {
            ERR_print_errors_fp(stderr);
            exit(-1);
        }

        printf("Connection made with [version,cipher]:
                [%s,%s]\n",SSL_get_version(my_ssl),SSL_get_cipher(my_ssl));

        for(wrote = 0; wrote < strlen(buffer); wrote += error) {
            error = SSL_write(my_ssl,buffer+wrote,strlen(buffer)-wrote);

            if(error <= 0)
                break;
        }

        SSL_shutdown(my_ssl);
        SSL_free(my_ssl);
    }

    SSL_CTX_free(my_ssl_ctx);
    SSL_BIO_free(server_bio);

    return 0;
}
```

Biblioteka OpenSSL

- Implementacja protokołów SSL/TSL
- Procedury kryptograficzne
- Generatory liczb losowych
- Wsparcie działań na wielkich liczbach

Inicjalizacja

```
OpenSSL_add_all_algorithms();
SSL_load_error_strings();
```

Struktury wykorzystywane przez bibliotekę

- SSL_METHOD – SSLv1, SSLv2, ..., TLSv1

```
SSL_METHOD *my_ssl_method;
my_ssl_method = TLSv1_method();
```

- SSL_CTX

```
SSL_CTX *my_ssl_ctx;
my_ssl_ctx = SSL_CTX_new(my_ssl_method)
```

```
if((my_ssl_ctx = SSL_CTX_new(my_ssl_method)) == NULL) {
     ERR_print_errors_fp(stderr);
     exit(1);
}
```

- SSL

```
SSL *my_ssl;
my_ssl = SSL_new(my_ssl_ctx
```

```
if((my_ssl = SSL_new(my_ssl_ctx)) == NULL) {
     ERR_print_errors_fp(stderr);
     exit(-1);
}
```

- BIO