TLS 통신

▶ OpenSSL: https_simple.c

```c
/*
 * MIT License
 *
 * Copyright (c) 2018 Lewis Van Winkle
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <errno.h>

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

#include <openssl/crypto.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/ssl.h>
#include <openssl/err.h>

#define ISVALIDSOCKET(s) ((s) >= 0)
#define CLOSESOCKET(s) close(s)
#define SOCKET int
#define GETSOCKETERRNO() (errno)
```

```c
int main(int argc, char *argv[]) {

    /*OpenSSL 사용 전 초기화*/
    SSL_library_init();

    //사용할 알고리즘 모두 불러오기
    OpenSSL_add_all_algorithms();

    //OpenSSL에서 에러 로드하기 -> 에러 처리에 이용
    SSL_load_error_strings();

    //SSL Conetxt 만들기 - 암호화키, 인증키 저장 후 connection동안 유지
    //SSL_CTX_new 파라미터로 하나만 들어갈 수 있음
    //TLS_client_method() -> TLS 사용한다..
    SSL_CTX *ctx = SSL_CTX_new(TLS_client_method());
    if (!ctx) {
        fprintf(stderr, "SSL_CTX_new() failed.\n");
        return 1;
    }

    if (argc < 3) {
        fprintf(stderr, "usage: https_simple hostname port\n");
        return 1;
    }

    char *hostname = argv[1];
    char *port = argv[2];

    /*TCP connection*/
    printf("Configuring remote address...\n");
    struct addrinfo hints;
    memset(&hints, 0, sizeof(hints));
    hints.ai_socktype = SOCK_STREAM;
    struct addrinfo *peer_address;
    //host -> ip
    if (getaddrinfo(hostname, port, &hints, &peer_address)) {
        fprintf(stderr, "getaddrinfo() failed. (%d)\n", GETSOCKETERRNO());
        exit(1);
    }

    printf("Remote address is: ");
    char address_buffer[100];
    char service_buffer[100];
    //ip -> host name
    getnameinfo(peer_address->ai_addr, peer_address->ai_addrlen,
            address_buffer, sizeof(address_buffer),
            service_buffer, sizeof(service_buffer),
            NI_NUMERICHOST);
    printf("%s %s\n", address_buffer, service_buffer);

    printf("Creating socket...\n");
    //서버 소켓 생성
    SOCKET server;
    server = socket(peer_address->ai_family,
            peer_address->ai_socktype, peer_address->ai_protocol);
    if (!ISVALIDSOCKET(server)) {
        fprintf(stderr, "socket() failed. (%d)\n", GETSOCKETERRNO());
        exit(1);
    }
```

```
    printf("Connecting...\n");
    //서버(host)에 연결
    if (connect(server,
                peer_address->ai_addr, peer_address->ai_addrlen)) {
        fprintf(stderr, "connect() failed. (%d)\n", GETSOCKETERRNO());
        exit(1);
    }
    freeaddrinfo(peer_address);

    printf("Connected.\n\n");
    /*TCP 연결 완료*/


    /*initiate TLS connection*/
    //SSL object 생성
    //만든 객체로 connection tracking 가능
    SSL *ssl = SSL_new(ctx);
    if (!ctx) {
        fprintf(stderr, "SSL_new() failed.\n");
        return 1;
    }

    //서버에 도메인 세팅
    //선택사항이지만, 없으면 여러 사이트에서 어떤 곳에 인증서를 보낼지 모름
    if (!SSL_set_tlsext_host_name(ssl, hostname)) {
        fprintf(stderr, "SSL_set_tlsext_host_name() failed.\n");
        ERR_print_errors_fp(stderr);
        return 1;
    }

    //TCP connection에 SSL 붙이기
    SSL_set_fd(ssl, server);
    if (SSL_connect(ssl) == -1) {
        fprintf(stderr, "SSL_connect() failed.\n");
        ERR_print_errors_fp(stderr);
        return 1;
    }
    /*TLS connection end*/

    //SSL_get_cipher(ssl) -> 세팅한 알고리즘 list 가져오기
    printf ("SSL/TLS using %s\n", SSL_get_cipher(ssl));


    //접속한 서버의 인증서 가져오기
    X509 *cert = SSL_get_peer_certificate(ssl);
    if (!cert) {
        fprintf(stderr, "SSL_get_peer_certificate() failed.\n");
        return 1;
    }

    char *tmp;
    //인증서 정보 가져오기
    if ((tmp = X509_NAME_oneline(X509_get_subject_name(cert), 0, 0))) {
        printf("subject: %s\n", tmp);
        OPENSSL_free(tmp);
    }
```

```
    if ((tmp = X509_NAME_oneline(X509_get_issuer_name(cert), 0, 0))) {
        printf("issuer: %s\n", tmp);
        OPENSSL_free(tmp);
    }

    X509_free(cert);


    char buffer[2048];

    sprintf(buffer, "GET / HTTP/1.1\r\n");
    sprintf(buffer + strlen(buffer), "Host: %s:%s\r\n", hostname, port);
    sprintf(buffer + strlen(buffer), "Connection: close\r\n");
    sprintf(buffer + strlen(buffer), "User-Agent: https_simple\r\n");
    sprintf(buffer + strlen(buffer), "\r\n");

    //SSL_write, SSL_read로 읽기 쓰기 수행가능
    SSL_write(ssl, buffer, strlen(buffer));
    printf("Sent Headers:\n%s", buffer);

    while(1) {
        int bytes_received = SSL_read(ssl, buffer, sizeof(buffer));
            if (bytes_received < 1) {
                printf("\nConnection closed by peer.\n");
                break;
            }

            printf("Received (%d bytes): '%.*s'\n",
                    bytes_received, bytes_received, buffer);

    } //end while(1)

    printf("\nClosing socket...\n");
    //연결 끊기
    SSL_shutdown(ssl);
    CLOSESOCKET(server);
    SSL_free(ssl);
    //할당 해제
    SSL_CTX_free(ctx);

    printf("Finished.\n");
    return 0;
}
```

1. SSL 초기화

```
/*OpenSSL 사용 전 초기화*/
SSL_library_init();

//사용할 알고리즘 모두 불러오기
OpenSSL_add_all_algorithms();

//OpenSSL에서 에러 로드하기 -> 에러 처리에 이용
SSL_load_error_strings();
```

2. SSL context 만들어서 TLS 선언

```
//SSL Conetxt 만들기 - 암호화키, 인증키 저장 후 connection동안 유지
//SSL_CTX_new 파라미터로 하나만 들어갈 수 있음
//TLS_client_method() -> TLS 사용한다..
SSL_CTX *ctx = SSL_CTX_new(TLS_client_method());
```

3. TCP connection 후, TLS connection 연결

```
/*initiate TLS connection*/
//SSL object 생성
//만든 객체로 connection tracking 가능
SSL *ssl = SSL_new(ctx);
if (!ctx) {
    fprintf(stderr, "SSL_new() failed.\n");
    return 1;
}

//서버에 도메인 세팅
//선택사항이지만, 없으면 여러 사이트에서 어떤 곳에 인증서를 보낼지 모름
if (!SSL_set_tlsext_host_name(ssl, hostname)) {
    fprintf(stderr, "SSL_set_tlsext_host_name() failed.\n");
    ERR_print_errors_fp(stderr);
    return 1;
}

//TCP connection에 SSL 붙이기
SSL_set_fd(ssl, server);
if (SSL_connect(ssl) == -1) {
    fprintf(stderr, "SSL_connect() failed.\n");
    ERR_print_errors_fp(stderr);
    return 1;
}
/*TLS connection end*/
```

4. 읽기 / 쓰기

```
SSL_write()
SSL_read()
```

5. 할당 해제

```
//연결 끊기
SSL_shutdown(ssl);
CLOSESOCKET(server);
SSL_free(ssl);
```

```
//할당 해제
SSL_CTX_free(ctx);
```

6. 사용한 알고리즘 리스트

```
SSL_get_cipher(ssl);
```

7. Cipher and Certificate

```
//접속한 서버의 인증서 가져오기
X509 *cert = SSL_get_peer_certificate(ssl);
if (!cert) {
    fprintf(stderr, "SSL_get_peer_certificate() failed.\n");
    return 1;
}

char *tmp;
//인증서 정보 가져오기
if ((tmp = X509_NAME_oneline(X509_get_subject_name(cert), 0, 0))) {
    printf("subject: %s\n", tmp);
    OPENSSL_free(tmp);
}

if ((tmp = X509_NAME_oneline(X509_get_issuer_name(cert), 0, 0))) {
    printf("issuer: %s\n", tmp);
    OPENSSL_free(tmp);
}

X509_free(cert);
```

8. complie

```
gcc https_simple.c -o https_simple -lssl -lcrypto
```

https 프로토콜을 이용하여 웹서버로부터 데이터를 가져오는 클라이언트 코드이다.

TLS 서버 구축

인증서를 직접 만들 수 있다.

- 직접 만든 것이기에 self-sign
- subject name == issure name 동일

▶ TLS Server

```c
//SSL-Server.c
#include <errno.h>
#include <unistd.h>
#include <malloc.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <resolv.h>
#include "openssl/ssl.h"
#include "openssl/err.h"

#define FAIL    -1

int OpenListener(int port)
{   int sd;
    struct sockaddr_in addr;

    sd = socket(PF_INET, SOCK_STREAM, 0);
    bzero(&addr, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = INADDR_ANY;
    if ( bind(sd, (struct sockaddr*)&addr, sizeof(addr)) != 0 )
    {
        perror("can't bind port");
        abort();
    }
    if ( listen(sd, 10) != 0 )
    {
        perror("Can't configure listening port");
        abort();
    }
    return sd;
}

SSL_CTX* InitServerCTX(void)
{
    const SSL_METHOD *method;
    SSL_CTX *ctx;

    //알고리즘, 에러 로드
    OpenSSL_add_all_algorithms();  /* load & register all cryptos, etc. */
    SSL_load_error_strings();   /* load all error messages */

    // method = TLSv1_2_server_method();
    method = TLS_server_method();

    ctx = SSL_CTX_new(method);   /* create new context from method */
    if(ctx == NULL)
    {
        ERR_print_errors_fp(stderr);
        abort();
    }

    //모든 알고리즘 준비됨
    SSL_CTX_set_cipher_list(ctx, "ALL:eNULL");

    return ctx;
}

//self-sign하는 과정
void LoadCertificates(SSL_CTX* ctx, char* CertFile, char* KeyFile)
{
    //New lines
    if (SSL_CTX_load_verify_locations(ctx, CertFile, KeyFile) != 1)
        ERR_print_errors_fp(stderr);

    if (SSL_CTX_set_default_verify_paths(ctx) != 1)
```

```
        ERR_print_errors_fp(stderr);
    //End new lines

    /* set the local certificate from CertFile */
    if (SSL_CTX_use_certificate_file(ctx, CertFile, SSL_FILETYPE_PEM) <= 0)
    {
        ERR_print_errors_fp(stderr);
        abort();
    }
    /* set the private key from KeyFile (may be the same as CertFile) */
    SSL_CTX_set_default_passwd_cb_userdata(ctx, "12345678");
    if (SSL_CTX_use_PrivateKey_file(ctx, KeyFile, SSL_FILETYPE_PEM) <= 0)
    {
        ERR_print_errors_fp(stderr);
        abort();
    }
    /* verify private key */
    if (!SSL_CTX_check_private_key(ctx))
    {
        fprintf(stderr, "Private key does not match the public certificate\n");
        abort();
    }

    //New lines - Force the client-side have a certificate
    //SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER | SSL_VERIFY_FAIL_IF_NO_PEER_CERT, NULL);
    //SSL_CTX_set_verify_depth(ctx, 4);
    //End new lines
}

void ShowCerts(SSL* ssl)
{
    X509 *cert;
    char *line;

    //인증서 정보 출력
    cert = SSL_get_peer_certificate(ssl); /* Get certificates (if available) */
    if ( cert != NULL )
    {
        printf("Server certificates:\n");
        line = X509_NAME_oneline(X509_get_subject_name(cert), 0, 0);
        printf("Subject: %s\n", line);
        free(line);
        line = X509_NAME_oneline(X509_get_issuer_name(cert), 0, 0);
        printf("Issuer: %s\n", line);
        free(line);
        X509_free(cert);
    }
    else
        printf("No certificates.\n");
}

//서버 통신
void Servlet(SSL* ssl) /* Serve the connection -- threadable */
{
    char buf[1024];
    int sd, bytes;

    char enter[3] = { 0x0d, 0x0a, 0x00 };

    char output[1024];
    strcpy(output, "HTTP/1.1 200 OK");
    strcat(output, enter);
    strcat(output, "Content-Type: text/html");
    strcat(output, enter);
    strcat(output, "Content-Length: 47");
    strcat(output, enter);
    strcat(output, enter);
    strcat(output, "<html><body><h1>Hello World!</h1></body></html>");

    if ( SSL_accept(ssl) == FAIL )     /* do SSL-protocol accept */
        ERR_print_errors_fp(stderr);
    else
    {
```

```
    {
        //인증시 체크용으로 출력
        ShowCerts(ssl);        /* get any certificates */
        //client가 request 암호화해서 보냄 -> 복호화하여 buf에 저장하기
        bytes = SSL_read(ssl, buf, sizeof(buf)); /* get request */
        if ( bytes > 0 )
        {
            buf[bytes] = 0;
            printf("Client msg: \"%s", buf);
            SSL_write(ssl, output, strlen(output)); /* send reply */
        }
        else
            ERR_print_errors_fp(stderr);
    }
    sd = SSL_get_fd(ssl);       /* get socket connection */
    SSL_free(ssl);         /* release SSL state */
    close(sd);          /* close connection */
}

int main(int argc, char **argv)
{
    SSL_CTX *ctx;
    int server;
    char portnum[]="5000";

    char CertFile[] = "key/certificate.crt";
    char KeyFile[] = "key/private_key.pem";

    SSL_library_init();

    ctx = InitServerCTX();        /* initialize SSL */
    LoadCertificates(ctx, CertFile, KeyFile); /* load certs */
    //port number로 listen
    server = OpenListener(atoi(portnum));    /* create server socket */
    while (1)
    {
        struct sockaddr_in addr;
        socklen_t len = sizeof(addr);
        SSL *ssl;

        //서버 요청 받음
        int client = accept(server, (struct sockaddr*)&addr, &len);  /* accept connection as usual */
        printf("Connection: %s:%d\n",inet_ntoa(addr.sin_addr), ntohs(addr.sin_port));
        //SSL 객체 생성
        ssl = SSL_new(ctx);              /* get new SSL state with context */
        SSL_set_fd(ssl, client);      /* set connection socket to SSL state */
        Servlet(ssl);         /* service connection */
    }
    close(server);          /* close server socket */
    SSL_CTX_free(ctx);         /* release context */
}
```

끝..