(https://aticleworld.com/)

(https://www.facebook.com/Aticleworld-602608163238897/)

(https://www.linkedin.com/in/amlendra-kumar-bb3b2096/)

(https://www.youtube.com/channel/UCAcrc6X4yzEjza9QXM7vg2A?sub_confirmation=1)

(https://twitter.com/aticleworld)

Home (https://aticleworld.com/)

C Tutorial (https://aticleworld.com/c-tutorial/)

C Programming Examples (https://aticleworld.com/c-programming/)

Interview Questions **Video Courses**

Tools (https://aticleworld.com/resources/)

Write for us (https://aticleworld.com/guest-article/)

,

ssl server client programming using openssl in C

BY **AMLENDRA (HTTPS://ATICLEWORLD.COM/AUTHOR/PRITOSH/)** ON **(HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/)**

30 (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENTS)

The Internet is like a sea, it’s open a lot of opportunities for the new world. There is a lot of company, which depend on the internet. The Internet reduces the workload and time of the people.

Now day’s people do not use the conventional way to send the information from one place to another place but using the internet they are sending the information. Previously people used the cash money for purchasing but nowadays they are using the internet for the purchasing.

So to making all the things secure which are transferring over the network, introduce a protocol SSL/TLS. It creates a secure connection between the client and the server.

What is SSL? (https://aticleworld.com/ssl-server-client-using-openssl-in-c/)

An SSL (Secure Sockets Layer) is the standard security protocol used to establish an encrypted connection between a server and a client. After establishing the connection SSL/TLS ensures that the data transmitted between server and client (https://aticleworld.com/socket-programming-in-c-using-tcpip/) are secured and intact.

SSL is used by many applications and banking websites to make the data private and secure. It provides security in the transmission of sensitive data like credit/debit card number, user login name, and password.

Note: A Good book for SSL/TLS, **"Bulletproof SSL and TLS" (https://geni.us/SSLBook)**

Working of SSL

SSL is designed to exchange sensitive data over the network using some secure algorithms and prevent from another program that wants to access the private data from the network connection.

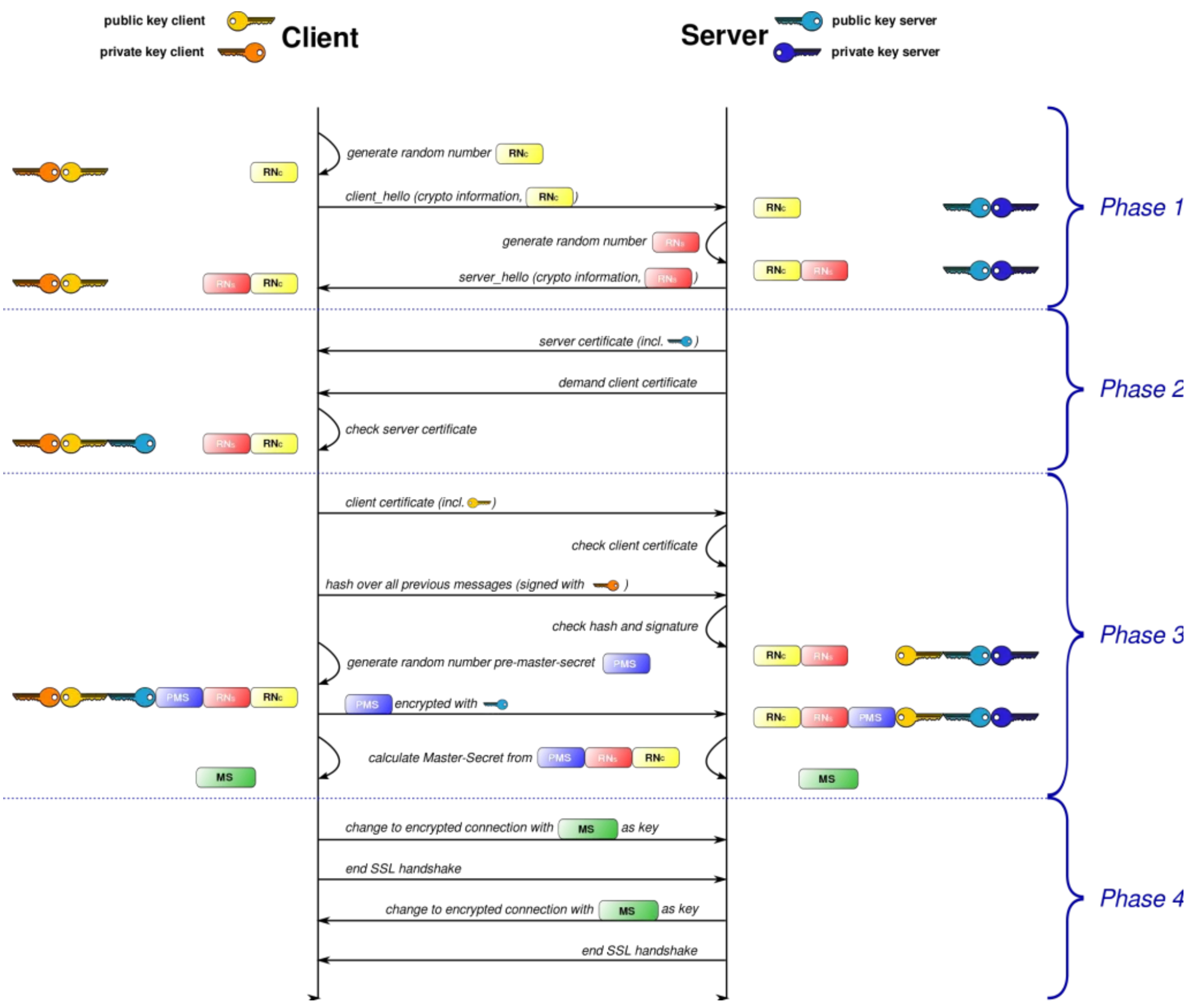
SSL uses asymmetric encryption algorithms to secure the transmission of data. These algorithms use the pair of keys (public and private). The public key is freely available and known for anybody. The private key is only known by the server or the client.In SSL data encrypted by the public key can only decrypt by the private key and the data encrypted by the private key can only decrypt by the public key.

In the SSL communication, the client starts the connection from the first hello (SSL) message. This hello message starts the negotiation and performs the handshaking between server and client. After completing the handshaking if everything is fine then generate a secured key for the current connection. The server and client have used this secret key in data exchanging.

SSL handshake flow (https://aticleworld.com/ssl-server-client-using-openssl-in-c/)

The SSL handshake is an authentication process. In which server and client authenticate to each other using a certificate.
This certificate is generated by the user own self with the help of OpenSSL commands or it is provided by a third party (certificate authority).

Below I am describing some steps which described the handshaking between the server and client.


public key client
private key client
Client
Server
public key server
private key server
generate random number RNc
client_hello (crypto information, RNc)
generate random number RNs
server_hello (crypto information, RNs)
Phase 1
server certificate (incl.)
demand client certificate
check server certificate
Phase 2
client certificate (incl.)
check client certificate
hash over all previous messages (signed with)
check hash and signature
generate random number pre-master-secret PMS
PMS encrypted with
calculate Master-Secret from PMS RNs RNc
MS
Phase 3
change to encrypted connection with MS as key
end SSL handshake
change to encrypted connection with MS as key
end SSL handshake
Phase 4

[(https://amzn.to/2BVaDgg)](https://amzn.to/2BVaDgg)

- In the beginning of the communication, SSL/TLS client sends a "client_hello" message to the server. This message contains all the cryptographic information which is supported by the client, like highest protocol version of SSL/TLS, encryption algorithm lists (in the client's order of preference), data compression method, resume session identifier and randomly generated data (which will be used in symmetric key generation).

- The SSL/TLS server responds with a “server_hello” message to give all the things which are required to establish a connection like protocol version used, data compression algorithms and encryption method selected, assigned session id and random data (which will be used in symmetric key generation).
- The server sends a certificate to the client and also insert a request message for the client certificate because server required the client certificate for the mutual authentication.
- The SSL or TLS client verifies the server’s digital certificate. For more information, see How SSL and TLS provide identification, authentication, confidentiality, and integrity.
- If the SSL or TLS server sent a “client certificate request”, the client sends a random byte string encrypted with the client’s private key, together with the client’s digital certificate, or a “no digital certificate alert”. This alert is only a warning, but with some implementations, the handshake fails if client authentication is mandatory.
- The SSL or TLS client sends the randomly generated data that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data. The randomly generated data itself is encrypted with the server’s public key.
- The SSL or TLS server verifies the client’s certificate.
- The SSL or TLS client sends the server a “finished” message, which is encrypted with the secret key, indicating that the client part of the handshake is complete.
- The SSL or TLS server sends the client a “finished” message, which is encrypted with the secret key, indicating that the server part of the handshake is complete.
- For the duration of the SSL or TLS session, the server and client can now exchange messages that are symmetrically encrypted with the shared secret key.

If you want to learn more about the TCP/IP, here 10 Free days (up to 200 minutes) TCP/IP video course (https://pluralsight.pxf.io/c/1192901/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Ftcp-ip-networking-for-devs) for you.

(https://pluralsight.pxf.io/c/1192901/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Ftcp-ip-networking-for-devs)

Example of secure server-client program using OpenSSL in C

In this example code, we will create a secure connection between client and server using the TLS1.2 protocol. In this communication, the client sends an XML request to the server which contains the username and password.

The server verifies the **XML request (https://aticleworld.com/create-an-xml-request-in-c-for-server-communication/)**, if it is valid then it sends a proper XML response to the client either give a message of Invalid Request.

Install the OpenSSL library, for the ubuntu use the below command.

sudo apt-get install libssl–dev

Before compiling the client and server program you will need a Certificate. You can generate your own certificate using the below command.

openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem

Note: *Here certificate name is mycert.pem.*

Example Client code for TLS1.2 communication

Compile the Client : gcc -Wall -o client Client.c -L/usr/lib -lssl -lcrypto

Run :   ./client <host_name> <port_number>

```
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <malloc.h>
#include <string.h>
#include <sys/socket.h>
#include <resolv.h>
#include <netdb.h>
#include <openssl/ssl.h>
#include <openssl/err.h>

#define FAIL    -1

int OpenConnection(const char *hostname, int port)
{
    int sd;
    struct hostent *host;
    struct sockaddr_in addr;

    if ( (host = gethostbyname(hostname)) == NULL )
    {
        perror(hostname);
        abort();
    }
    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 = *(long*)(host->h_addr);
    if ( connect(sd, (struct sockaddr*)&addr, sizeof(addr)) != 0 )
    {
        close(sd);
        perror(hostname);
        abort();
    }
    return sd;
}

SSL_CTX* InitCTX(void)
{
    SSL_METHOD *method;
    SSL_CTX *ctx;

    OpenSSL_add_all_algorithms();  /* Load cryptos, et.al. */
    SSL_load_error_strings();   /* Bring in and register error messages */
    method = TLSv1_2_client_method();  /* Create new client-method instance */
    ctx = SSL_CTX_new(method);   /* Create new context */
    if ( ctx == NULL )
    {
        ERR_print_errors_fp(stderr);
        abort();
    }
    return ctx;
}

void ShowCerts(SSL* ssl)
{
    X509 *cert;
    char *line;
    cert = SSL_get_peer_certificate(ssl); /* get the server's certificate */
    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);       /* free the malloc'ed string */
        line = X509_NAME_oneline(X509_get_issuer_name(cert), 0, 0);
        printf("Issuer: %s\n", line);
        free(line);       /* free the malloc'ed string */
        X509_free(cert);     /* free the malloc'ed certificate copy */
    }
    else
        printf("Info: No client certificates configured.\n");
}

int main(int count, char *strings[])
{
    SSL_CTX *ctx;
    int server;
    SSL *ssl;
    char buf[1024];
    char acClientRequest[1024] = {0};
    int bytes;
    char *hostname, *portnum;

    if ( count != 3 )
    {
        printf("usage: %s <hostname> <portnum>\n", strings[0]);
        exit(0);
    }
    SSL_library_init();
    hostname=strings[1];
    portnum=strings[2];

    ctx = InitCTX();
    server = OpenConnection(hostname, atoi(portnum));
    ssl = SSL_new(ctx);      /* create new SSL connection state */
    SSL_set_fd(ssl, server);    /* attach the socket descriptor */
    if ( SSL_connect(ssl) == FAIL )   /* perform the connection */
        ERR_print_errors_fp(stderr);
    else
    {

        char acUsername[16] = {0};
        char acPassword[16] = {0};
        const char *cpRequestMessage = "<Body>\
                               <UserName>%s<UserName>\
                 <Password>%s<Password>\
                 <\Body>";

        printf("Enter the User Name : ");
        scanf("%s",acUsername);

        printf("\n\nEnter the Password : ");
        scanf("%s",acPassword);

        sprintf(acClientRequest, cpRequestMessage, acUsername,acPassword);   /* cc

        printf("\n\nConnected with %s encryption\n", SSL_get_cipher(ssl));
        ShowCerts(ssl);        /* get any certs */
        SSL_write(ssl,acClientRequest, strlen(acClientRequest));   /* encrypt & se
        bytes = SSL_read(ssl, buf, sizeof(buf)); /* get reply & decrypt */
        buf[bytes] = 0;
        printf("Received: \"%s\"\n", buf);
        SSL_free(ssl);         /* release connection state */
    }
    close(server);          /* close socket */
    SSL_CTX_free(ctx);         /* release context */
    return 0;
}
```

(https://shareasale.com/r.cfm?b=768058&u=1658724&m=59485&urllink=&afftrack=)

Example Server code for TLS1.2 communication

Compile the Server : gcc -Wall -o server Server.c -L/usr/lib -lssl -lcrypto

Run : sudo ./server <portnum>

```
#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



// Create the SSL socket and intialize the socket address structure
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;
}

int isRoot()
{
    if (getuid() != 0)
    {
        return 0;
    }
    else
    {
        return 1;
    }

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

    OpenSSL_add_all_algorithms();  /* load & register all cryptos, etc. */
    SSL_load_error_strings();   /* load all error messages */
    method = TLSv1_2_server_method();  /* create new server-method instance */
    ctx = SSL_CTX_new(method);   /* create new context from method */
    if ( ctx == NULL )
    {
        ERR_print_errors_fp(stderr);
        abort();
    }
    return ctx;
}

void LoadCertificates(SSL_CTX* ctx, char* CertFile, char* KeyFile)
{
    /* 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) */
    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();
    }
}

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

    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");
}

void Servlet(SSL* ssl) /* Serve the connection -- threadable */
{
    char buf[1024] = {0};

    int sd, bytes;
    const char* ServerResponse="<\Body>\
                               <Name>aticleworld.com</Name>\
                 <year>1.5</year>\
                 <BlogType>Embedede and c\c++<\BlogType>\
                 <Author>amlendra<Author>\
                 <\Body>";

    const char *cpValidMessage = "<Body>\
                               <UserName>aticle<UserName>\
                 <Password>123<Password>\
                 <\Body>";

    if ( SSL_accept(ssl) == FAIL )     /* do SSL-protocol accept */
        ERR_print_errors_fp(stderr);
    else
    {
        ShowCerts(ssl);        /* get any certificates */
        bytes = SSL_read(ssl, buf, sizeof(buf)); /* get request */
        buf[bytes] = '\0';

        printf("Client msg: \"%s\"\n", buf);

        if ( bytes > 0 )
        {
            if(strcmp(cpValidMessage,buf) == 0)
            {
                SSL_write(ssl, ServerResponse, strlen(ServerResponse)); /* send re
            }
            else
            {
                SSL_write(ssl, "Invalid Message", strlen("Invalid Message")); /* s
            }
        }
        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 count, char *Argc[])
{
    SSL_CTX *ctx;
    int server;
    char *portnum;


//Only root user have the permsion to run the server
    if(!isRoot())
    {
        printf("This program must be run as root/sudo user!!");
        exit(0);
    }
    if ( count != 2 )
    {
        printf("Usage: %s <portnum>\n", Argc[0]);
        exit(0);
    }

    // Initialize the SSL library
    SSL_library_init();

    portnum = Argc[1];
    ctx = InitServerCTX();        /* initialize SSL */
    LoadCertificates(ctx, "mycert.pem", "mycert.pem"); /* load certs */
    server = OpenListener(atoi(portnum));    /* create server socket */
    while (1)
    {
        struct sockaddr_in addr;
        socklen_t len = sizeof(addr);
        SSL *ssl;

        int client = accept(server, (struct sockaddr*)&addr, &len);  /* accept con
        printf("Connection: %s:%d\n",inet_ntoa(addr.sin_addr), ntohs(addr.sin_port
        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 */
}
```

How to run client-server program?

Server run first, using the below command we will run the server and wait for the client request.

```
sudo ./server  8081
```

Note: *In above command 8081 is the port number.*

After that, we will run client using the below command and send the XML request.

```
./client  127.0.0.1 8081
```

Note: *In above command, 127.0.0.1 is the local host IP and 8081 is the port number.*

If the client sends a valid request as per the server then server gives a proper response.

Client XML Request (https://aticleworld.com/create-an-xml-request-in-c-for-server-communication/):

```
"<Body>
<UserName>aticle</UserName>
<Password>123</Password>
</Body>"
```

Server Response:

```
"<Body>
<Name>aticleworld.com</Name>
<year>1.5</year>
<BlogType>Embedede and c c++</BlogType>
<Author>amlendra</Author>
</Body>"
```

If the client sends an invalid request to the server then server give a response to an "Invalid message".

Client XML Request (https://aticleworld.com/create-an-xml-request-in-c-for-server-communication/):

```
"<Body>
<UserName>amlendra</UserName>
<Password>1235</Password>
</Body>"
```

Server Response:

"Invalid Message"

Recommended Post

- **Transport Layer Security (TLS) (https://aticleworld.com/transport-layer-security-tls/)**
- **Socket programming in C (https://aticleworld.com/socket-programming-in-c-using-tcpip/)**.
- **HDLC Protocol in C. (https://aticleworld.com/hdlc-protocol/)**
- **Difference between HDLC and PPP. (https://aticleworld.com/difference-between-hdlc-and-ppp/)**
- **Parse XML response in C without using the library. (https://aticleworld.com/parse-xml-response-in-c/)**
- **Create Http Get and Post request in C (https://aticleworld.com/http-get-and-post-methods-example-in-c/)**.
- **File handling in C (https://aticleworld.com/file-handling-in-c/)**.
- **I2C Communication protocol (https://aticleworld.com/i2c-bus-protocol-and-interface/)**.
- **Embedded C Interview Questions. (https://aticleworld.com/embedded-c-interview-questions-2/)**
- **Pointers in C. (https://aticleworld.com/pointers-in-c/)**
- **CAN Protocol Interview Questions. (https://aticleworld.com/can-protocol-interview-questions-and-answers-in-detail/)**
- **Bit-wise interview Questions in C. (https://aticleworld.com/interview-questions-on-bitwise-operators-in-c/)**

References: **http://www.cs.utah.edu/~swalton/listings/sockets/programs/**

(http://www.cs.utah.edu/~swalton/listings/sockets/programs/)

THIS ENTRY WAS POSTED IN **C LANGUAGE (HTTPS://ATICLEWORLD.COM/CATEGORY/C-LANGUAGE/)**, **COMMUNICATION PROTOCOL (HTTPS://ATICLEWORLD.COM/CATEGORY/COMMUNICATION-PROTOCOL/)**. BOOKMARK THE **PERMALINK (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/)**.

About Amlendra (https://aticleworld.com/author/pritosh/)

I am an embedded c software engineer and a corporate trainer, currently, I am working as senior software engineer in a largest Software consulting company . I have working experience of different microcontrollers (stm32, LPC, PIC AVR and 8051), drivers (USB and virtual com-port), POS device (VeriFone) and payment gateway (global and first data).

WEBSITE (HTTPS://ATICLEWORLD.COM/)

← PREVIOUS ARTICLE (HTTPS://ATICLEWORLD.COM/SOCKET-PROGRAMMING-IN-C-USING-TCPIP/)

NEXT ARTICLE (HTTPS://ATICLEWORLD.COM/INHERITANCE-IN-C/) →

(https://aticleworld.com/socket-programming-in-c-using-tcpip/)

Socket programming in c using TCP/IP (https://aticleworld.com/socket-programming-in-c-using-tcpip/)

AMLENDRA (HTTPS://ATICLEWORLD.COM/AUTHOR/PRITOSH/)/ (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/)

(https://aticleworld.com/inheritance-in-c/)

Inheritance in c++ (https://aticleworld.com/inheritance-in-c/)

AMLENDRA (HTTPS://ATICLEWORLD.COM/AUTHOR/PRITOSH/)/ (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/)

30 comments

Radu

JANUARY 4, 2018 AT 6:20 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-750)

The servers is displaying No certificates. SSL_get_peer_certificates only returns a certificate on the server side if the client has send a certificate. But the code does not request a client certificate which means that the client will not send one. How do we fix this?

REPLY

GIORGIO BIRRA

SEPTEMBER 24, 2019 AT 10:55 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-3449)

Check the copy from browser in your editor. I have the some problem but during the copy, my pc misses this row in client source:

sprintf(acClientRequest, cpRequestMessage, acUsername,acPassword); /* construct reply */

REPLY

GIORGIO BIRRA

SEPTEMBER 24, 2019 AT 10:56 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-3450)

Sorry, i had the same error, now it works ☺

REPLY

ansh

JANUARY 13, 2018 AT 7:53 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-751)

can you please with no certificates message. i need to implement 2 way tls, any pointers would be helpful.

REPLY

Schwab

FEBRUARY 22, 2018 AT 1:17 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-778)

Hey everybody, has someone solut the Problem with the certificates??

REPLY

Farid

MARCH 12, 2018 AT 11:45 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-788)

Thank you for nice article . How can i send simple GET request to tls secured web site . For example :

GET
https://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20weather.forecast%20where%20woeid%20in%20(select%20woeid%20from%20geo.places(1)%20where%20text%3D%22nome%2C%20ak%22)&format=json&env=store%3A%2F%2Fdatatables.org%2Falltableswithkeys (https://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20weather.forecast%20where%20woeid%20in%20(select%20woeid%20from%20geo.places(1)%20where%20text%3D%22nome%2C%20ak%22)&format=json&env=store%3A%2F%2Fdatatables.org%2Falltableswithkeys)

REPLY

Amlendra

MARCH 12, 2018 AT 11:53 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-789)

See this article, might be helpful. https://aticleworld.com/http-get-and-post-methods-example-in-c/ (https://aticleworld.com/http-get-and-post-methods-example-in-c/)

REPLY

Farid

MARCH 13, 2018 AT 6:41 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-791)

The sample that you post is for http requests . But i Need to read data from secured channel .
Here is the question and sample that i am using . Du you have any idea why that code is not working ?
https://stackoverflow.com/questions/49195088/c-tls-ssl-read-delays-and-returns-0-bytes (https://stackoverflow.com/questions/49195088/c-tls-ssl-read-delays-and-returns-0-bytes)

REPLY

Amlendra

MARCH 14, 2018 AT 11:09 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-792)

The link is not working.

REPLY

Vicente Gimeno

MARCH 20, 2018 AT 11:29 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-795)

Just for clarification: you need to generate or obtain your own certificates if you want the server to succeed.

REPLY

Richard Wicks

JUNE 6, 2018 AT 3:44 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-996)

OK, I went ahead and did this. There's some very minor errors in the code, but they were all in the strings. I cleaned it up and checked it into github – if anybody would like it.

https://github.com/mrwicks/miscellaneous/tree/master/tls_1.2_example (https://github.com/mrwicks/miscellaneous/tree/master/tls_1.2_example)

Also, to the original author, if you want me to remove this from github, let me know, and I will. That's just a site for me to access code I've been repeatedly rewriting over time – well, some of it.

REPLY

Kranti MadineniKranti

JUNE 25, 2018 AT 5:15 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1011)

Thank You Very Much, Sir.

REPLY

Amlendra (https://aticleworld.com/)

JUNE 26, 2018 AT 11:13 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1012)

Welcome.

REPLY

Rahul

JULY 7, 2018 AT 6:43 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1028)

What if we like to access particular URL via proxy using openssl? How we can do that

REPLY

Richard Wicks

JULY 26, 2018 AT 2:58 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1046)

Proxy is an entirely different thing to deal with. If you're just trying to grab a webpage with a command line tool, use wget. I believe that will handle it.

REPLY

Anil

OCTOBER 14, 2018 AT 8:06 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1209)

thanks sir …

REPLY

sreenadh

OCTOBER 18, 2018 AT 11:03 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1219)

Hi Amlendra…, nice article…, but mutual certification not done here…, can you give client certification code. code does not request a client certificate which means that the client will not send one. How do we fix this?

REPLY

sreenadh

OCTOBER 18, 2018 AT 11:05 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1220)

this code not having client certificate which means that the client will not send one. How do we fix this?

REPLY

---

mohan

OCTOBER 27, 2018 AT 8:14 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1241)

sir please help me ssl mongoose websocket. I have tried with cesnta example its not working for me. wss is not handshake done. do you have any worked code with mongoose pls send to me.

REPLY

---

l84Inch

OCTOBER 31, 2018 AT 9:09 AM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1250)

Request/verify of a client cert is controlled by mode settings in the SSL_CTX.
(ref – https://www.openssl.org/docs/man1.0.2/ssl/SSL_CTX_set_verify.html (https://www.openssl.org/docs/man1.0.2/ssl/SSL_CTX_set_verify.html)). Probably inInitServerCTX() you want to add something like:
…
int mode = SSL_VERIFY_PEER |
…
SSL_CTX_set_verify(ctx, mode, NULL);
return ctx;
…
(no, I haven't complied/tested the above, but you get the idea).
The above page also includes a sample verification routine (replaces NULL above), which seems to be the right place to go pawing through any certificate that the client presents.

REPLY

Michael Uman

NOVEMBER 18, 2018 AT 11:34 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-1299)

Why do you call the message and reply XML? If anything it is very badly formed XML. Well formed XML has open and close tags. value…

Otherwise I like the ssl code. Thx

REPLY

IoTiAn

DECEMBER 11, 2019 AT 2:29 PM (HTTPS://ATICLEWORLD.COM/SSL-SERVER-CLIENT-USING-OPENSSL-IN-C/#COMMENT-3766)

I am created my own mqtt server with ssl feature but when i try to connect with client to server it shows the following error:
SSL_read(2) – error reading data
Dropping client

REPLY

Pingback: http get and post methods example in c - AticleWorld (https://aticleworld.com/http-get-and-post-methods-example-in-c/)

Pingback: Parse XML response in C without using library - AticleWorld (https://aticleworld.com/parse-xml-response-in-c/)

Pingback: Difference Between High-level Data Link Control (HDLC) and Point-to-Point Protocol (PPP) - AticleWorld (https://aticleworld.com/difference-between-hdlc-and-ppp/)

Pingback: Transport Layer Security (TLS) - AticleWorld (https://aticleworld.com/transport-layer-security-tls/)

Pingback: Active, Reactive and Apparent Power - AticleWorld (https://aticleworld.com/active-reactive-and-apparent-power/)

Pingback: Understanding Linear Regression - AticleWorld (https://aticleworld.com/understanding-linear-regression/)

Pingback: Difference between Active and Reactive Power (Active vs Reactive) - AticleWorld (https://aticleworld.com/difference-between-active-and-reactive-power/)

Pingback: Metaboly (https://blog.barakinnovations.in/2020/08/ssl-server-client-programming-using-openssl-in-c/)

Leave a Reply

Enter your comment here...

(https://aticleworld.com/best-c-programming-books/)

(https://aticleworld.com/best-gift-programmers/)

(https://www.udemy.com/?ranMID=39197&ranEAID=UajS5soDmWY&ranSiteID=UajS5soDmWY-ZAN8mpepGLirkhLP5qf9Nw&LSNPUBID=UajS5soDmWY)

Join Aticleworld

You will also get our free C interview questions eBook

Enter your email here*

Subscribe

Pages

About (https://aticleworld.com/about/)
Guest Article (https://aticleworld.com/guest-article/)
Blog Posts (https://aticleworld.com/blog-post/)
affiliate-disclosure (https://aticleworld.com/affiliate-disclosure/)
disclaimer (https://aticleworld.com/disclaimer/)

Categories

8051 Micro-controller (https://aticleworld.com/category/8051-micro-controller/)
Autocad (https://aticleworld.com/category/autocad/)
batch file (https://aticleworld.com/category/batch-file/)
Blogging tips (https://aticleworld.com/category/blogging-tips/)

C Language (https://aticleworld.com/category/c-language/)
C# Language (https://aticleworld.com/category/c-language-2/)
C++ Language (https://aticleworld.com/category/c/)
communication protocol (https://aticleworld.com/category/communication-protocol/)
Data science (https://aticleworld.com/category/data-science/)
Electrical (https://aticleworld.com/category/electrical/)
File handling (https://aticleworld.com/category/c-language/file-handling/)
Linux (https://aticleworld.com/category/linux/)
Operating System (https://aticleworld.com/category/operating-system/)
PIC microcontroller (https://aticleworld.com/category/pic-microcontroller/)
Professional Certificates (https://aticleworld.com/category/professional-certificates/)
Python (https://aticleworld.com/category/python/)
Review (https://aticleworld.com/category/review/)
Rtos (https://aticleworld.com/category/rtos/)
Star patterns in C (https://aticleworld.com/category/c-language/star-patterns-in-c/)
Uncategorized (https://aticleworld.com/category/uncategorized/)
Windows Driver (https://aticleworld.com/category/windows-driver/)

BACK TO TOP ↑