Turn a simple socket into an SSL socket

Asked 11 years, 11 months ago Modified 1 year, 1 month ago Viewed 181k times

139

I wrote simple C programs, which are using sockets ('client' and 'server'). (UNIX/Linux usage)

The server side simply creates a socket:

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

And then binds it to sockaddr:

```
bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
```

And listens (and accepts and reads):

```
listen(sockfd,5);
newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
read(newsockfd,buffer,255);
```

The client creates the socket, and then writes to it.

Now, I want to convert this simple connection into an SSL connection, in the plainest, most idyllic, neatest and quickest way.

I've tried to add OpenSSL to my project, but I can't find an easy way to implement what I want.

c linux sockets unix ssl

Share Follow

edited Jun 23, 2016 at 17:28

Peter Mortensen
30.8k ● 21 ● 105 ● 131

asked Oct 8, 2011 at 17:19

David Mape
1,407 ● 2 ● 10 ● 3

If you're looking for "a secure connection" rather than SSL in particular, you could look at something like proxychains.sourceforge.net which resides outside your application, and set that up to send traffic over an SSH connection. As far as in-application SSL, OpenSSL is pretty easy if you understand how SSL/TLS is supposed to work. If you want an alternative, try yaSSL or gnuTLS . – Borealid Oct 8, 2011 at 17:23

3 Define 'easy way'. OpenSSl is the standard for C programmers. If you're having difficulty with it you should ask about that. – user207421 Apr 25, 2013 at 1:57

Check this one An Introduction to OpenSSL Programming (Par t I). Part II is too advanced and difficult for me. But part2 is worth taking a look. – Rick Nov 21, 2019 at 14:18

Also check Secure programming with the OpenSSL API. But I just heard opinions about how bad Openssl is and other alternatives worth a try. – Rick Nov 21, 2019 at 15:08

1 Another option is to use an external SSL wrapper tool such as `stunnel`, the `stunnel4` package is in Debian-based distros and it's easy to use. There are some limitations compared to adding proper SSL support in your server, but it can be good for a quick solution. I like stunnel because it seems to fit with the UNIX software tools approach. – Sam Watkins Jul 5, 2020 at 5:06

5 Answers

Sorted by: Highest score (default)

185

+125

There are several steps when using OpenSSL. You must have an SSL certificate made which can contain the certificate with the private key be sure to specify the exact location of the certificate (this example has it in the root). There are a lot of good tutorials out there.

- Some documentation and tools from HP (see chapter 2)
- Command line for OpenSSL

Some includes:

```
#include <openssl/applink.c>
#include <openssl/bio.h>
#include <openssl/ssl.h>
#include <openssl/err.h>
```

You will need to initialize OpenSSL:

```
void InitializeSSL()
{
    SSL_load_error_strings();
    SSL_library_init();
    OpenSSL_add_all_algorithms();
}

void DestroySSL()
{
    ERR_free_strings();
    EVP_cleanup();
}

void ShutdownSSL()
{
    SSL_shutdown(cSSL);
    SSL_free(cSSL);
}
```

Now for the bulk of the functionality. You may want to add a while loop on connections.

```
int sockfd, newsockfd;
SSL_CTX *sslctx;
SSL *cSSL;

InitializeSSL();
```

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd< 0)
{
    //Log and Error
    return;
}
struct sockaddr_in saiServerAddress;
bzero((char *) &saiServerAddress, sizeof(saiServerAddress));
saiServerAddress.sin_family = AF_INET;
saiServerAddress.sin_addr.s_addr = serv_addr;
saiServerAddress.sin_port = htons(aPortNumber);

bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));

listen(sockfd,5);
newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);

sslctx = SSL_CTX_new( SSLv23_server_method());
SSL_CTX_set_options(sslctx, SSL_OP_SINGLE_DH_USE);
int use_cert = SSL_CTX_use_certificate_file(sslctx, "/serverCertificate.pem" ,
SSL_FILETYPE_PEM);

int use_prv = SSL_CTX_use_PrivateKey_file(sslctx, "/serverCertificate.pem",
SSL_FILETYPE_PEM);

cSSL = SSL_new(sslctx);
SSL_set_fd(cSSL, newsockfd );
//Here is the SSL Accept portion.  Now all reads and writes must use SSL
ssl_err = SSL_accept(cSSL);
if(ssl_err <= 0)
{
    //Error occurred, log and close down ssl
    ShutdownSSL();
}
```

You are then able read or write using:

```
SSL_read(cSSL, (char *)charBuffer, nBytesToRead);
SSL_write(cSSL, "Hi :3\n", 6);
```

Update The `SSL_CTX_new` should be called with the TLS method that best fits your needs in order to support the newer versions of security, instead of `SSLv23_server_method()`. See: [OpenSSL SSL_CTX_new description]

> *TLS_method(), TLS_server_method(), TLS_client_method().* These are the general-purpose *version-flexible* SSL/TLS methods. The actual protocol version used will be negotiated to the highest version mutually supported by the client and the server. The supported protocols are SSLv3, TLSv1, TLSv1.1, TLSv1.2 and TLSv1.3.

Share Follow

edited Dec 13, 2018 at 17:14

answered May 1, 2013 at 23:36

CaptainBli

4,121 ● 4 ● 39 ● 58

12 not so "simple" as I thought, but finally (thanks God!) I see some code. Is this cross-platform or just for unix/unix-like systems? – juliomalegria May 1, 2013 at 23:55

3 I have used similar code on multiple platforms: arm, linux and windows. – CaptainBli Sep 15, 2013 at 16:11

2 Last `if` is wrong though. It should be `if (ssl_err <= 0) {` only then it's an error. `SSL_accept()` returns `1` on success, `0` on "controlled failure" and `-1` on "fatal failure". See man page. – Jite Aug 25, 2014 at 12:02

2 Also, DH ciphers will not work if `SSL_CTX_set_tmp_dh[_callback]()` is not called. Just discovered the hard way that aNULL ciphers won't work without it, producing alert number 40. – Roman Dmitrienko Jul 19, 2015 at 15:37

6 @DevNull The `SSLv23_server_method()` states that the server understands SSLv2 and v3 and is now deprecated. To support TLS 1.1 and 1.2 replace that method with `TLS_server_method()`. source – ezPaint Jun 23, 2016 at 15:21

22

OpenSSL is quite difficult. It's easy to accidentally throw away all your security by not doing negotiation exactly right. (Heck, I've been personally bitten by a bug where curl wasn't reading the OpenSSL alerts exactly right, and couldn't talk to some sites.)

If you really want quick and simple, put stud in front of your program an call it a day. Having SSL in a different process won't slow you down: http://vincent.bernat.im/en/blog/2011-ssl-benchmark.html

Share Follow

answered Apr 26, 2013 at 3:40

BraveNewCurrency
12.7k ● 2 ● 42 ● 50

17 This is a practical answer, but it doesn't really answer the question. – Swiss May 1, 2013 at 23:50

6 STUD was abandoned in 2016. The readme recommends: github.com/varnish/hitch – Charles Dec 26, 2016 at 20:54

11

For others like me:

There was once an example in the SSL source in the directory `demos/ssl/` with example code in C++. Now it's available only via the history:
https://github.com/openssl/openssl/tree/691064c47fd6a7d11189df00a0d1b94d8051cbe0/demos/ssl

You probably will have to find a working version, I originally posted this answer at Nov 6 2015. And I had to edit the source -- not much.

Certificates: .pem in `demos/certs/apps/`:
https://github.com/openssl/openssl/tree/master/demos/certs/apps

Share Follow

edited Jul 12, 2019 at 18:11

Captain Man
7,017 ● 6 ● 48 ● 74

answered Nov 6, 2015 at 13:34

18446744073709551615
16.4k ● 4 ● 94 ● 127

0

Like you've seen it's pretty hairy. But the "easy" part is, after you've set up your SSL/TLS session with OpenSSL, the pattern you follow to read/write on a socket for HTTPS is the same as the one you follow for HTTP.

The difference is you use the `SSL_read/SSL_write` functions, instead of the `read/write` functions. The `SSL_read/SSL_write` functions take an `SSL` pointer as their first argument instead of a file descriptor.

Anyways, here is a full working OpenSSL example for a Unix environment, with compile/run instructions, APT dependencies, and references. It's just some more working code for you to cut your teeth on.

On successful compilation, you will see one warning about a deprecated OpenSSL function.

On successful run, you will see example.com's TLS certificate subject printed to standard output, followed by the HTML content of example.com.

https://github.com/angstyloop/c-web/blob/main/openssl-fetch-example.c

Tested on Ubuntu 22.04.

Share Follow

answered Aug 14, 2022 at 7:24

angstyloop
117 ● 6

-3

Here my example ssl socket server threads (multiple connection)
https://github.com/breakermind/CppLinux/blob/master/QtSslServerThreads/breakermindsslserver.cpp

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <unistd.h>
#include <iostream>

#include <breakermindsslserver.h>

using namespace std;

int main(int argc, char *argv[])
{
    BreakermindSslServer boom;
    boom.Start(123,"/home/user/c++/qt/BreakermindServer/certificate.crt",
"/home/user/c++/qt/BreakermindServer/private.key");
    return 0;
}
```

Share Follow

answered Nov 8, 2017 at 11:06

Fanex
1

Please include the solution directly in your SO post. – Maciej Jureczko Nov 8, 2017 at 11:26

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.