

EECS 498 W26 System Design of a Search Engine

09 LinuxGetSsl

Due Sunday, February 22, 2026, 11:59 pm

In this assignment, you will read webpages with HTTPS by completing the code in the LinuxGetSsl utility.

Starter files

The starter files in LinuxGetUrlSslStarter.zip are the same as for the previous assignment.

The assignment

For HTTPS, our connection will be wrapped by SSL/TLS, using port 443. This allows the entire request and response to be encrypted. Both the server and client will agree on a public key mechanism and a secret session key that they will use to communicate with before they send any data. Other than that, the process itself is identical to HTTP.

After we connect to an HTTPS server, but before we send our GET request, we need to wrap our connection with OpenSSL. First, we must initialize OpenSSL. (Note: The Autograder requires you to explicitly setup OpenSSL, but newer versions of OpenSSL do not).

```
SSL_library_init( );
```

After initializing OpenSSL, an SSL context needs to be created which specifies our connection method with the server. (Note: TLS_method() is the most widely used method, but the Autograder only supports SSLv23_method(). For this project, use SSLv23_method(), but be aware that this is deprecated in later versions of OpenSSL).

```
SSL_CTX *ctx = SSL_CTX_new( SSLv23_method( ) );
```

With our method of communication, we will initialize a new ssl connection, and assign a socket to wrap around. The SSL objects need to know which socket they're wrapping around, as the underlying socket is the method of sending and receiving encrypted data.

```
SSL *ssl = SSL_new( ctx );  
SSL_set_fd( ssl, socketFd );
```

Finally, we need to connect with the server using a TLS handshake. The client and the server work out the details of the connections on their own through OpenSSL.

```
SSL_connect( ssl );
```

Reading and writing with SSL

To read and write with SSL, we need to use functions from the OpenSSL library, to keep the data encrypted. Our new read and write functions are:

```
SSL_read( ssl, buffer, len );  
SSL_write( ssl, buffer, len );
```

Cleaning up SSL

Finally, we need to clean up and free our added SSL resources. To close our connection with the server and free our SSL context, we can do:

```
SSL_shutdown( ssl );  
SSL_free( ssl );  
close( sockfd );  
SSL_CTX_free( ctx );
```

That's it! You now have a functional HTTP/S website reader!

Autograder

Submit your LinuxGetSsl.cpp file to the autograder at <https://autograder.io/web/project/3496>.

Enhancements

This is a bare bones reader. Your search engine will probably need an enhanced version that also supports:

1. Chunked transfers. Some websites use chunked transfer encoding. Your crawler should still be able to read these web pages.
2. Gzip'ed content. Some websites will respond with content that's compressed with gzip. To support this, you'll need to install zlib (sudo apt install zlib1g-dev) and compile with -lz to link with zlib.

3. Redirects. Some URLs will redirect you when you request content, e.g., <https://google.com> redirects to <https://www.google.com>.

FAQ

Frequently asked questions on why the program is not working:

Why am I getting a 400 Invalid Request?

Make sure your lines in your GET request are delimited by `\r\n`.

Why am I getting a 408 Request Timeout?

Make sure your GET request ends with a `\r\n\r\n`.

OpenSSL does not work for Xcode?

Following these steps might help you solve your problem:

Install OpenSSL (can be done with `brew install openssl`). Note the file paths specified in the output following "For compilers to find openssl you may need to set:". Specifically, keep note of the library path (will look something like `/usr/local/opt/openssl/lib`) and the include path (will look something like `/usr/local/opt/openssl/include`).

1. Go into your Xcode project build settings. This can be done by clicking on the `xcodeproj` icon in the Project navigator and then selecting "Build Settings".
2. Find the "Header Search Paths" field and add a search path for the include path
3. Find the "Library Search Paths" field and add a search path for the library path
4. Find the "Other Linker Flags" field and add the flags `-lssl` and `-lcrypto`