LINUX JOURNAL

An Introduction to OpenSSL Programming, Part I of II

Sep 01, 2001 By Eric Rescorla (/user/800894)

in

Do you have a burning need to build a simple web client and server pair? Here's why OpenSSL is for you.

(/issue/89)

From Issue #89 September 2001 (/issue/89)

The quickest and easiest way to secure a TCP-based network application is with SSL. If you're working in C, your best choice is probably to use OpenSSL (http://www.openssl.org/ (http://www.openssl.org)). OpenSSL is a free (BSD-style license) implementation of SSL/TLS based on Eric Young's SSLeay package. Unfortunately, the documentation and sample code distributed with OpenSSL leave something to be desired. Where they exist, the manual pages are pretty good, but they often miss the big picture, as manual pages are intended as a reference, not a tutorial.

The OpenSSL API is vast and complicated, so we won't attempt to provide anything like complete coverage here. Rather, the idea is to teach you enough to work effectively from the manual pages. In this article, the first of two, we will build a simple web client and server pair that demonstrate the basic features of OpenSSL. In the second article, we will introduce a number of advanced features, such as session resumption and client authentication.

I assume that you're already familiar with SSL and HTTP, at least at a conceptual level. If you're not, a good place to start is with the RFCs (see Resources).

For space reasons, this article only includes excerpts from the source code. The complete source code is available in machine-readable format from the author's web site at http://www.rtfm.com/openssl-examples/ (http://www.rtfm.com/openssl-examples) .

Programs

Our client is a simple HTTPS (see RFC 2818) client. It initiates an SSL connection to the server and then transmits an HTTP request over that connection. It then waits for the response from the server and prints it to the screen. This is a vastly simplified version of the functionality found in programs like fetch and cURL.

The server program is a simple HTTPS server. It waits for TCP connections from clients. When it accepts one it negotiates an SSL connection. Once the connection is negotiated, it reads the client's HTTP request. It then transmits the HTTP response to the client. Once the response is transmitted it closes the connection.

Our first task is to set up a context object (an SSL_CTX). This context object is then used to create a new connection object for each new SSL connection. These connection objects are used to do SSL handshakes, reads and writes.

This approach has two advantages. First, the context object allows many structures to be initialized only once, improving performance. In most applications, every SSL connection will use the same keying material, certificate authority (CA) list, etc. Rather than reloading this material for every connection, we simply load it into the context object at program startup. When we wish to create a new connection, we can simply point that connection to the context object. The second advantage of having a single context object is that it allows multiple SSL connections to share data, such as the SSL session cache used for session resumption. Context

initialization consists of four primary tasks, all performed by the initialize_ctx() function, shown in Listing 1.

Listing 1. initialize_ctx() (/files/linuxjournal.com/linuxjournal/articles/048/4822/4822l1.html)

Before OpenSSL can be used for anything, the library as a whole must be initialized. This is accomplished with SSL_library_init(), which primarily loads up the algorithms that OpenSSL will be using. If we want good reporting of errors, we also need to load the error strings using SSL_load_error_strings(). Otherwise, we won't be able to map OpenSSL errors to strings.

We also create an object to be used as an error-printing context. OpenSSL uses an abstraction called a BIO object for input and output. This allows the programmer to use the same functions for different kinds of I/O channels (sockets, terminal, memory buffers, etc.) merely by using different kinds of BIO objects. In this case we create a BIO object attached to stderr to be used for printing errors.

If you're writing a server or a client that is able to perform client authentication, you'll need to load your own public/private key pair and the associated certificate. The certificate is stored in the clear and is loaded together with the CA certificates forming the certificate chain using SSL_CTX_use_certificate_chain_file(). SSL_CTX_use_PrivateKey_file() is used to load the private key. For security reasons, the private key is usually encrypted under a password. If it is, the password callback (set using SSL_CTX_set_default_passwd_cb()) will be called to obtain the password.

If you're going to be authenticating the host to which you're connected, OpenSSL needs to know what CAs you trust. The SSL_CTX_load_verify_locations() call is used to load the CAs.

In order to have good security, SSL needs a good source of strong random numbers. In general, it's the responsibility of the application to supply some seed material for the random number generator (RNG). However, OpenSSL automatically uses /dev/urandom to seed the RNG, if it is available. Because /dev/urandom is standard on Linux, we don't have to do anything for this, which is convenient because gathering random numbers is tricky and easy to screw up. Note that if you're on a system other than Linux, you may get an error at some point because the random number generator is unseeded. OpenSSL's rand(3) manual page provides more information.

Comments

how can check certificate against CRL in Linux using SSL command (/article/4822#comment-339418)

Submitted by laxman (not verified) on Jul 09, 2009.

Hi,

How can client checks certificate against CRL(certification revocation List) using SSL libray functions.

I have loaded certifcate using X509_load_cert_crl_file(pLookup,"crl.pem",X509_FILETYPE_PEM);

but i am not getting how client can checks....

Thanks & Regards,
Laxman

Great article, need some info (/article/4822#comment-320560)
Submitted by saleem (not verified) on Apr 03, 2008.

Hi,
Your article is simple and yet very informative.
Thanks for the article.

I am new to openssl, i am stuck with a problem. I need some info.

Due to some reasons the cert info is available in a buffer rather than a file.
My client needs to load this and establish communication with the server using openssl.

The buffer looks something like this....
char *gacacert =
"MIICLzCCAiswggGUoAMCAQICBgEYgSDT3DANBgkqhkiG9w0BAQUFADA0MRAwDgYD\n\
VQQKEwdlbnRydXN0MQwwCgYDVQQLEwNlbmcxEjAQBgNVBAMTCWdhTG9jYWxDQTAe\n\
Fw0wODAzMDUyMjQ3MzVaFw0yODAyMjkyMjQ3MzVaMDQxEDAOBgNVBAoTB2VudHJ1\n\
c3QxDDAKBgNVBAsTA2VuZzESMBAGA1UEAxMJZ2FMb2NhbENBMIGfMA0GCSqGSIb3\n\
DQEBAQUAA4GNADCBiQKBgQDW4ONrqPZ/Hc9Ft/vL1eD76XpbxhdmAezpjGK0aWa2\n\
2QCkDD6IpU3VxpW93+i8em2zgCV5fujbcJuNebk+Y24q3w8FVbba7BZGcaoatB99\n\
vdZ0gp/t/DXq9KsdxdlE2W/mKBCvxkkMsEnm5kHeHZXByouqPvlXGBsJORCH2ahB\n\
vwIDAQABo0gwRjASBgNVHRMBAf8ECDAGAQH/AgEAMBEGCWCGSAGG+EIBAQQEAwIA\n\
JDAdBgNVHQ4EFgQUIZVCc+92iSwt3CD3P9TYIJB6pLQwDQYJKoZIhvcNAQEFBQAD\n\
gYEAjZq3mZ/Q6F26BBd74Q5lJcABGTM4nB1mThaCJk//dLx6WhmWoXJoZD0//nYM\n\
UDvISCc4KtMZoe5qkO/BKJs9IwsXQyZiPl5bAtcfN6OmSe+fmNPMUKD1ck8l7WLu\n\
7k6hlBwrIli05KhiYLY5i4ZbVh0+DyjIkXbv2GJj+g0CrEE=";

Could some one tell me how to load this buffer, communicate with the server and perform the handshake?

Any code sample/example will be great.

Thanks,
Saleem

@ saleem Hi I am facing (/article/4822#comment-344862)
Submitted by Rakesh (not verified) on Nov 02, 2009.

@ saleem

Hi

I am facing with the same issue.

I tried using this function to create a mem BIO which can be used to read from the memory.
BIO_new_mem_buf((char *){YOUR STRING},-1);

I was able to read the string from the certificate. But the problem was that when i pass if to SSL_CTX_use_certificate_chain_file it fails.

Please send me any info if you have found a solution already

Thanks
Rakesh

SSLcontext with BSD Sockets (/article/4822#comment-252868)
Submitted by Xtremejames183 (not verified) on Jun 05, 2007.

Hi all,
Great article but i want to know how to attach SSL Context to BSD sockets(socket(),bind(),listen(),connect()..)
Thanks for help

Nice work, Eric. Thank (/article/4822#comment-252588)
Submitted by Anonymous (not verified) on Jun 04, 2007.

Nice work, Eric. Thank you.

Just a comment: You looks like John Petrucci (guitarist of Dream Theater band)

Thanks for your nice (/article/4822#comment-193185)
Submitted by duanjigang (not verified) on Nov 01, 2006.

Thanks for your nice artical, i am learning it

freeing of returned resource in check_cert() (/article/4822#comment-154868)
Submitted by manaz (not verified) on Aug 04, 2006.

nice article, but there is a little memory leak in the check_cert() function. pointer returned from SSL_get_peer_certificate must be freed according to man page:

The reference count of the X509 object is incremented by one, so that it will not be destroyed when the session containing the peer certificate is freed. The X509 object must be explicitly freed using X509_free().

Re: An Introduction to OpenSSL Programming, Part I of II (/article/4822#comment-937)
Submitted by Anonymous on Mar 24, 2004.

Nice article

need more info (/article/4822#comment-30306)
Submitted by Student (not verified) on Apr 15, 2005.

can u please send me the link of where the next part is ?? that is part 2 of 2 ... thanks for the wonderful tutorial

Re: An Introduction to OpenSSL Programming, Part I of II (/article/4822#comment-938)
Submitted by Anonymous on Oct 18, 2004.

huh!! nice.....

very good , makes it look all (/article/4822#comment-121690)
Submitted by Anonymous (not verified) on Jul 28, 2005.

very good , makes it look all so easy! now to put it into practice

Post new comment

Subject:

Comment: *

Allowed HTML tags: <a>   <cite> <code> <pre>   <dl> <dt> <dd> <i> <b>
Lines and paragraphs break automatically.
Use to create page breaks.

More information about formatting options (/filter/tips)

Preview