Secure programming with the OpenSSL API

Create basic secure and unsecure connections

Level: Intermediate

Kenneth Ballard (kenneth.ballard@ptk.org)
Freelance programmer
22 Jul 2004

Learning how to use the API for OpenSSL -- the best-known open library for secure communication -- can be intimidating, because the documentation is incomplete. Fill in the gaps, and tame the API, with the tips in this article. After setting up a basic connection, see how to use OpenSSL's BIO library to set up both a secured and unsecured connection. And learn a bit about error detection as well.

Related content:

Network programming with the Twisted framework, Part 4
Programming Linux sockets, Part 1
Programming Linux sockets, Part 2
Communications Programming Concepts: Sockets
Introduction to cryptography
Understanding Sockets in Unix, NT, and Java

Subscriptions:

dW newsletters
dW Subscription (CDs and downloads)

The documentation to the OpenSSL API is a little vague. Not many tutorials on the use of OpenSSL exist either, so getting it to work in applications can be a little troublesome for beginners. So how can you implement a basic secure connection using OpenSSL? This guide will help to solve that problem.

Part of the problem with learning how to implement OpenSSL is the fact that the documentation is not complete. An incomplete API documentation normally keeps developers from using the API, which normally spells doom for it. Yet OpenSSL is still around and going strong. Why?

OpenSSL is the best-known open library for secure communication. A Google search for "SSL library" returns OpenSSL at the top of the list. It started life in 1998 being derived from the SSLeay library developed by Eric Young and Tim Hudson. Other SSL toolkits include GNU TLS, distributed under the GNU General Public License, and Mozilla Network Security Services (NSS) (see Resources later in this article for additional information).

So what makes OpenSSL better than GNU TLS, Mozilla NSS, or any other library? Licensing is one issue (see Resources). In addition, GNS TLS (thus far) supports only TLS v1.0 and SSL v3.0 protocols, and not much more.

Mozilla NSS is distributed under both the Mozilla Public License and the GNU GPL, allowing the developer to pick. But Mozilla NSS is larger than OpenSSL and requires other external libraries to build the library, whereas OpenSSL is entirely self-contained. And like OpenSSL, much of the NSS API is not documented. Mozilla NSS has PKCS #11 support, which is used for cryptographic tokens, such as Smart Cards. OpenSSL lacks this support.

Prerequisites

To get the most out of this article, you should:

- Be proficient in C programming
- Be familiar with Internet communication and writing Internet-enabled applications

A familiarity with SSL is not absolutely required, as a short explanation of SSL will be given later; however, look in the Resources section if you want to find links to articles discussing SSL in detail. A knowledge of cryptography is a plus as well, but not required.

What is SSL?

SSL is an acronym that stands for Secure Sockets Layer. It is the standard behind secure communication on the Internet, integrating data cryptography into the protocol. The data is encrypted before it even leaves your computer, and is decrypted only once it reaches its intended destination. Certificates and cryptographic algorithms are behind how it all works, and with OpenSSL, you have the opportunity to play around with both.

In theory, if the encrypted data were intercepted or eavesdropped before reaching its destination, there is no hope of cracking that data. But as computers become ever faster as each year passes, and new advances in cryptanalysis are made, the chance of cracking the cryptography protocols used in SSL is starting to increase.