MatrixSSL Readme

Description

MatrixSSL is an embedded, open source SSL implementation designed for small footprint applications and devices. It is designed to reduce the complexity of integrating SSL into an embedded project. With a simple API and security layer, users are able to easily integrate MatrixSSL with their applications. MatrixSSL uses industry standard cryptographic algorithms and protocols to ensure users are getting a strong and reliable security solution in an open-source package that is under 50K compiled.

MatrixSSL was designed to allow users to easily add support for new operating systems, crypto providers, and cipher suites. The package comes with built in support for Windows and Linux and cipher suites defined in the PeerSec embedded cryptography implementation.

Commercial Version

Some features described in this document are available only in the commercially licensed version of MatrixSSL. Sections of this document that refer to the commercial version will be noted.

Overview

A functional SSL implementation consists of two primary components: A handshake protocol between a client and server to securely negotiate a communication session over which secure messages will be transported, and a set of cryptographic algorithms used to secure the messages sent and received over that negotiated session.

MatrixSSL's handshake protocol supports the client and server side SSLv3.

The commercial version supports the TLS standard as well. These standard protocols are compatible with any client software that supports SSL such as Web browsers, Web servers and email clients.

MatrixSSL supports the SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA and SSL_RSA_WITH_3DES_EDE_CBC_SHA cipher suites.
The commercial version also supports the TLS_RSA_WITH_AES_128_CBC_SHA and TLS_RSA_WITH_AES_256_CBC_SHA cipher suites.
RSA is used as the public key encryption mechanism for authentication and key exchange. RSA keys are stored on disk in the PEM format. 3DES encrypted private RSA key files are supported as well. Message authentication codes are either MD5 or SHA1. 3DES and arc4 are used for encryption and decryption of messages.
The commercial version adds AES support for encryption and decryption.

PeerSec Networks has included an implementation of these cryptographic algorithms as part of the MatrixSSL distribution. MatrixSSL has been designed to allow users to easily

add or replace cipher suites, add or replace individual cryptographic algorithms, and allows easy integration of any crypto provider.

Installing

The MatrixSSL package does not include an installation program or script. Simply extract the supplied directory structure from the distribution media to a local file system.

Building MatrixSSL

This section explains how to build the MatrixSSL shared library and example httpsReflector server application. For development information on application integration, porting, implementing new cipher suites, and other compile-time configurations see the MatrixSSL Developers Guide.

Quick Start for Windows

These steps will enable a user to build the MatrixSSL library and an example server application very quickly. These steps assume a Windows box that has been configured with Visual Studio .NET.

1. Copy the MatrixSSL distribution package to a dedicated directory.
2. Open the httpsReflector.sln solution file located in the examples directory. This solution contains two projects: matrixssl, the project that builds the MatrixSSL library and httpsReflector, which builds an example application that uses MatrixSSL.
3. Make sure the httpsReflector project is the default project by right clicking the project name and selecting ‘Set As StartUp Project’.
4. Make sure the httpsReflector project has a dependency on the matrixssl project by right clicking the solution name and selecting ‘Project Dependencies’ and confirming the build order.
5. Build the solution through the 'Build' menu. You can specify Release or Debug targets through the Visual Studio configuration manager.
6. You can now run the httpsReflector.exe through the debugger or by double clicking the executable file. The httpsReflector application is a simple server application that accepts an HTTPS connection and echoes the data back to the client.
7. Open a web browser and connect to the server at https://localhost:4433/
8. Depending on the browser and security configuration options, a security dialog box should appear asking if you would like to proceed to the secure site. The sample certificates that are used will likely cause the dialog box to indicate the certificate is issued from a certificate authority you have not chosen to trust and that the IP address of the certificate does not match. This is normal for test certificates. Choosing to continue on to the page should produce a browser page similar to the following.

```
PeerSec Networks
Successful MatrixSSL request:
```

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, application/x-shockwave-flash, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
.NET CLR 1.0.3705; .NET CLR 1.1.4322)
Host: localhost:4433
Connection: Keep-Alive
```

9. The httpsClient solution may be built in a similar manner and executed from inside the debugger or standalone. The output from running the httpsClient against the httpsReflector should produce results similar to the following:

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
100 connections in 1 seconds (100.000000 c/s)

100 requests in 1 seconds (100.000000 r/s)
Press return to exit...
```

Quick Start for Linux

These steps will enable a user to create the MatrixSSL library and example server application very quickly. These steps assume a Linux box that has been configured with standard development tools (gcc compiler and *make*).

1. Copy the MatrixSSL distribution package to a dedicated directory.
2. Type ‘make’ from the src directory to build the MatrixSSL library.
3. Type ‘make’ from the examples directory to build the example server.
4. Run the example server application by typing ‘./httpsReflector’. The httpsReflector application is a simple server application that accepts an HTTPS connection and echoes the data back to the client.
5. Open a web browser and connect to the server at https://localhost:4433
6. Depending on the browser and security configuration options, a security dialog box should appear asking if you would like to proceed to the secure site. The sample certificates that are used will likely cause the dialog box to indicate the certificate is issued from a certificate authority you have not chosen to trust and that the IP address of the certificate does not match. This is normal for test certificates. Choosing to continue on to the page should produce a browser page similar to the following.

```
PeerSec Networks
Successful MatrixSSL request:
GET / HTTP/1.1
Host: localhost:4433
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.5a)
Gecko/20030728 Mozilla Firebird/0.6.1
Accept: text/xml, application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,video/x-
mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
```

```
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: Keep-Alive
```

7. The httpsClient solution may be built in a similar manner and executed from the command line. The output from running the httpsClient against the httpsReflector should produce results similar to the following:

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
100 connections in 1 seconds (100.000000 c/s)

100 requests in 1 seconds (100.000000 r/s)
Press return to exit...
```

MatrixSSL Directory structure

The top level directories in the full MatrixSSL distribution are *src*, *doc*, and *examples*. In addition, the public header for the library is located at the top level (*matrixSsl.h*).

The *src* directory (with the addition of the public header file) contains all the of the source code necessary to build the MatrixSSL library. The C code and header files at the top level of *src* are the core files of the product and should always be included when compiling.

The *os* and *crypto* subdirectories contain the 'pluggable' portions of MatrixSSL that may vary depending on the specific operating system and crypto provider being used. The provided default crypto provider in the *crypto* directory is *peersec* (the company that wrote MatrixSSL). Normally the built-in PeerSec crypto provider is sufficient, but if you need specific hardware implementations please contact PeerSec Networks for an easy to use OpenSSL crypto wrapper.

The supported operating systems supplied by default in the *os* directory are Windows and Linux (*win* and *linux)*. Additional OS ports can be added at this level.

The *examples* directory contains reference applications that use the MatrixSSL library. The *httpsReflector* example is a server that simply echoes a HTTPS request back to a client such as a Web browser. The *httpsClient* example implements client side HTTPS functionality and can interact securely with *httpsReflector* or another secure server such as Apache with OpenSSL.

The *doc* directory contains all the documentation for the MatrixSSL product.

1.2.4 Release Notes

- There was no public 1.2.3 release
- Functional changes from 1.2.2 release
  - Client will reply with NULL cert message if client authentication is requested.
- No API changes from 1.2.2 release
- Bug fixes and enhancements
  - Changed all instances of *int* types to *int32* to be more explicit and to allow easy global redefinitions for porting
  - Corrected the maximum message size limit to match the SSL specification
  - Developers may notice some internal routines using a psPool_t parameter. These parameters allow deterministic memory support in the commercial version of MatrixSSL. They are unused in the GNU version of MatrixSSL.
  - Cert parse can handle duplicate distinguished name entries.
  - ASN.1 parse fix for AlgorithmIdentifier missing the trailing NULL
  - Checking certificate version before doing checking the 'ca' member of the basic constraint entry during certificate validation.
- For a full list of release notes and issues, see http://www.matrixssl.org