MatrixSSL 3.1 Open Source Release Notes

PeerSec Networks, Inc. 410 Broadway Ave E. #205 Seattle, WA 98102 **T** 425.646.7850 **F** 206.501.4366

support@peersec.com www.peersec.com

Overview

Thank you for choosing MatrixSSL. The 3.1 version is a major revision to the previous releases and enables users to implement strong SSL security into their applications faster than ever. With a design emphasis on further reducing memory usage and providing an easier integration API, MatrixSSL 3.1 is the security solution for virtually any networked application on any platform.

If you are upgrading from a previous version of MatrixSSL you will want to read the MatrixSSL Developers Guide to learn more about the specific changes.

Who Is This Document For?

- Software developers that are securing applications with MatrixSSL
- Software developers upgrading to MatrixSSL 3.1 from a previous version
- Anyone wanting to learn more about MatrixSSL 3.1

New in 3.1

MatrixSSL 3.1 is the result of starting with the widely deployed PeerSec SSL/TLS engine and optimizing for reduced memory usage, cryptographic speed, and ease-of-integration. The following list is an overview of the most significant changes to this major revision.

TLS 1.0 Protocol Support

Beginning in MatrixSSL 3.1 the TLS 1.0 protocol and AES cipher are now available in open source releases.

Improved API

It is now easier than ever to integrate SSL into your application. MatrixSSL has always provided SSL integration to applications at a data buffer level to guarantee support for any given transport mechanism. Previous versions, however, left the management of these data buffers in the hands of the integrator. The new MatrixSSL 3.1 API incorporates size-optimized buffer management so the user is left only with the task of determining when data needs to be read or written, while still maintaing a transport-neutral, zero buffer copy API.

Faster and smaller RSA cryptography

The public key cryptography operations required for RSA mathematics are the primary contributors to high water memory and CPU resources during the SSL handshake. MatrixSSL 3.1 includes specific optimizations that have resulted in major improvements to both speed and memory usage during public cryptography. These substantial memory savings and performance improvements allow MatrixSSL to be used on an even larger number of embedded platforms. The entire SSL handshake, including network buffers can now be completed in as little as 10KB of RAM, with a post-handshake dynamic memory footprint of less than 3KB.

File and functional re-organization

The MatrixSSL 3.1 source code package has been organized to better reflect the individual functional areas. The *core* and *crypto* modules are now clear building blocks on which

MatrixSSL relies and each module has an API and Configuration header to manage optional features and functionality.

Supported Client and Server Applications

New client and server examples are now provided as a starting off point for customer integration or new application development. The client application is an example of a simple, blocking sockets API HTTPS client that prints the response to a HTTP GET request. The server example demonstrates a non-blocking HTTPS server that handles multiple connections and session timeouts. The MatrixSSL API usage for both applications is very similar, and should help clarify how to integrate MatrixSSL with other applications.

Self-Test Application

A SSL/TLS protocol test application is now included in the package so that new ports of MatrixSSL can quickly be verified and functionally tested, even before integration with a sockets layer. The application creates virtual SSL connections within a single process using memory buffers as the transport layer. Each supported cipher suite and handshake mode are validated.

Additional Project Formats

Project files for the MatrixSSL library, example and test applications are now provided for Microsoft Visual Studio Express Edition, Apple Xcode and standard GNU make. Projects for the Eclipse IDE can be directly imported from GNU Makefile.

Support and Bug Reporting

Known Issues

The current release must be compiled in 32 bit pointer mode (-m32 flag for GCC) on 64 bit platforms. 64 bit integers are supported in this release, so cryptography performance is not negatively affected.

Contacting Support or Reporting Bugs

Email support@peersec.com