MatrixSSL 3.1.1 Open Source Release Notes

Overview

MatrixSSL 3.1.1 Release Notes

Features and Functionality

Source Code Changes to the Public Interface

Bug Fixes

Support and Bug Reporting

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 migrating from a previous version of MatrixSSL you will want to read the Migrating To MatrixSSL 3 document 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 3.1.1 from a previous version
- Anyone wanting to learn more about MatrixSSL 3.1

MatrixSSL 3.1.1 Release Notes

This section highlights the differences between version 3.1 and 3.1.1

Changes to Features and Functionality

Secure Renegotiations

In late 2009 an exploit in the SSL renegotiation protocol was discovered. A fix for the exploit has been published in RFC 5746 and version 3.1.1 includes the implementation. Re-handshaking is now controlled by three new compile-time defines:

ENABLE_SECURE_REHANDSHAKES

This define can be found in *matrixsslConfig.h* and is enabled by default. Enabling this define will activate the RFC 5746 implementation and allow MatrixSSL applications to securely re-handshake with peers that have also implemented it.

REQUIRE_SECURE_ REHANDSHAKES

This define can be found in *matrixsslConfig.h* and is disabled by default. Enabling this define will only allow the MatrixSSL application to connect with SSL peers that have implemented RFC 5746.

ENABLE_INSECURE_ REHANDSHAKES

This define can be found in matrixsslConfig.h and is disabled by default. Enabling this define will enable legacy re-handshaking support and is NOT RECOMMENDED.

CLIENT_HELLO extension support

Support for adding extensions to CLIENT_HELLO messages is now included in the open source version of MatrixSSL. More information on hello extensions can be found in RFC 3546. To support this feature an existing API has changed prototype and three new APIs have been introduced:

matrixSslNewClientSession

This function prototype has changed. A new function callback parameter has been added to this routine that will be invoked while clients are parsing SERVER_HELLO extensions.

matrixSslNewHelloExtension, matrixSslLoadHelloExtension, and matrixSslDeleteHelloExtension

This family of APIs is now available to client application integrators to append CLIENT_HELLO extensions to the handshake protocol. See the API documentation for details on these new function.

Client cipher suites on re-handshakes

Clients will now resend the full list of supported cipher suites on server-initiated re-handshakes. In previous versions, upon receiving a HELLO_REQUEST from a connected server, the client would only supply the cipher suite that was currently negotiated in the CLIENT_HELLO.

Source Code Changes to the Public Interface

New matrixSslNewClientSession prototype

An additional parameter has been added to this routine to improve hello extension support. Clients can now register a callback that will be invoked during the SSL handshakes to parse any SERVER_HELLO extensions that might be sent by the server.

USE_INT64 renamed to HAVE_NATIVE_INT64

This define in *coreConfig.h* has been renamed for clarity.

Bug Fixes

Changing Cipher Suites on Re-handshake

A handshaking failure was discovered during re-handshake testing in some cases where the underlying cipher suite was changing. This has been fixed and there are no known issues with re-handshaking.

Default size for pstm_digit

The default 32-bit platform now explicitly sets the `psmt_digit` type as a 32-bit unsigned integer rather than an `unsigned long`. This fixes the problem of running with 32-bit math on a 64-bit platform.

Support and Bug Reporting

Known Issues

None

Contacting Support or Reporting Bugs

Email support@peersec.com