MatrixSSL 3.7.1

Open Source Release Notes

1 OVERVIEW

The previous MatrixSSL open source version was 3.6.2. This document highlights the changes from the 3.6.2 version.

There have been a few prototype changes for existing APIs. Those are outlined in the API Changes section below and discussed in detail in the MatrixSSL API document included in this product package.

2 New Features and Improvements

2.1 Run-Time TLS Feature Control

Truncated HMAC use, Maximum Fragment Length requests, and Elliptic Curve specification can now be enabled on a per-session basis when creating a new session. A new `sslSessOpts_t` structure has been added to the `matrixSslNewClientSession` and `matrixSslNewServerSession` APIs to enable the per-session run-time control.

The `USE_TRUNCATED_HMAC` and `REQUESTED_MAX_PLAINTEXT_RECORD_LEN` defines no longer exist now that these features are run-time rather than compile-time.

For details on the new `sslSessOpts_t` structure see the Session Options section in the API documentation.

2.2 Platform Detection Moved to Header Files

A set of platform tests has been moved from the *common.mk* Makefile build system into the *osdep.h* header file. This change allows the Visual Studio and Xcode projects to become simpler and to ease the creation of other custom build systems.

2.3 X.509 RSASSA-PSS Signatures

MatrixSSL now supports the more secure RSASSA-PSS signature algorithm in X.509 certificates.

2.4 Application-Layer Protocol Negotiation

RFC 7301 defines a TLS extension that allows clients and servers to negotiate the application protocol during the TLS handshake. This feature is intended to save any round trips that were required after the TLS connection to negotiate the protocol.

The new `USE_ALPN` define in *matrixsslConfig.h* enables the feature and a new client side API and a new server side API have been added to quickly integrate the feature into applications.

2.5 Opaque User Pointers

The `ssl_t` structure now has a `void *userPtr` member for application specific usage. The `ssl_t` member may be set during session creation using the `sslSessOpts_t` input or at any other time during the `ssl_t` lifecycle. The application specific pointer is intended to help identify sessions when application callbacks have been invoked by the MatrixSSL library.

The opaque pointer is now also passed to the core routines `psGetTime`, `psDiffMsecs`, and `psGetEntropy` to provide per-session context.

2.6 Session Context to Deterministic Memory APIs

The psMalloc family of APIs now enable the user to associate a custom opaque pointer to the psPool_t structure for per-session context in the memory module. The psFree and psRealloc APIs have changed prototypes to accommodate this feature. There are also a handful of public APIs that have changed prototype to support this feature and are listed in the API Changes section below.

2.7 Constant-Time Memory Compare Routine

Calls to POSIX C memcmp have been replaced with a memcmpct implementation that will not leak any timing information to the caller.

2.8 X.509 and ASN.1 Parsing Improvements

A security audit revealed a handful of parsing issues related to boundary testing. These have been fixed.

2.9 Windows Client Example Fixed

A getopt implementation has been added for Windows builds that enables the example client to compile cleanly out of the box.

3 API CHANGES

3.1 matrixSslNewClientSession and matrixSslNewServerSession

The final "int32 flags" parameters to the routines have been replaced by a `sslSessOpts_t` pointer that allows servers and clients to implement run-time control over some features that were previously compile-time. For example, maximum fragment length negotiations and truncated HMAC negotiations are now run-time configurations.

For details on the new `sslSessOpts_t` structure see the Session Options section in the API documentation.

3.2 psGetTime, psDiffMsecs, and psGetEntropy Changes

A new `void *userPtr` input parameter has been added to these core platform functions. The pointer will be whatever the user has set as the `ssl_t->userPtr` value. See the Opaque User Pointer section above for related information.

These changes only apply to users that have custom osdep.c layers for their platform. If using the default POSIX or WIN32 implementation, the change will be

3.3 matrixSslNewKeys, matrixSslNewSessionId, matrixSslLoadCRL, and matrixSslNewHelloExtension

These functions have added a final void *poolUserPtr parameter to enable callers to pass a custom opaque pointer to the underlying memory pools. This is only relevant to users of the Matrix Deterministic Memory feature who are implementing custom allocation routines. Existing implementations will simply add NULL as the final parameter to these routines.

See the Matrix Deterministic Memory document for details.

3.4 Addition of "const" Keyword

Some APIs have added the C language "const" qualifier to pointer parameters to help indicate the underlying implementation does not modify the input.

4 Frameworks Changes

4.1 PRNG Gathering

The gathering of PRNG data has been moved from the *matrixssl* module to the *crypto* module. This forces all consumers of PRNG for the platform to access random data from a single source to allow resource locking on platforms that require that. This change is transparent to the user.

4.2 Deterministic Memory Changes

All `psMalloc` related APIs now take a `psPool_t` pointer as an input parameter to enable custom implementations to create a context for each memory allocation and free. This change is transparent to any user not using the `psMalloc` functions in their application code.