MatrixSSL 3.6.1 Open Source Release Notes

TABLE OF CONTENTS

1 FEATURE ADDITIONS 3
1.1 TLS 1.2 3
1.2 ECC Cipher Suites 3
1.3 Diffie-Hellman Cipher Suites 3
1.4 AES_GCM Cipher Suites 3
1.5 Pre-Shared Key Cipher Suites 3
1.6 ServerNameIndication Hello Extension 4
1.7 Stateless Session Tickets 4
1.8 Truncated HMAC Extension 4
1.9 AES-NI for Supported Linux Intel Processors 4
1.10 SEED and IDEA Symmetric Ciphers 4
1.11 Client May Choose Subset of Cipher Suites 4
1.12 Choose TLS Protocol Version on a Per-Session Basis 4
1.13 Disable Re-Handshake Support on a Per-Session Basis 5
1.14 ZLIB Compression Support 5

2 API CHANGES 6
2.1 matrixSslNewClientSession 6
2.2 matrixSslNewServerSession 6
2.3 matrixSslReceivedData 6

3 BUG FIXES AND IMPROVEMENTS 7
3.1 Explicit Length Testing in Parsing Code 7
3.2 Server Session Cache Improvement 7
3.3 Additional X.509 Certificate Authentication Testing 7
3.4 Makefile Framework Improvement 7
3.5 Enhanced Example Client and Server Applications 7
3.6 Added 64-bit Div128 For Platforms Missing Internal Capability 7
3.7 NO_RENEGOTIATION Alert Now at Proper Warning Level 7
3.8 Continued X.509 Certificate Parsing Improvements 7
3.9 Fragmentation Support for CERTIFICATE_REQUEST Message 8
3.10 Included the HIGHRES_TIME Code for Testing 8

1 FEATURE ADDITIONS

MatrixSSL 3.6.0 Open Source introduces many TLS features that have previously only been available in commercial versions of the software. This list highlights the new features.

1.1 TLS 1.2

This latest TLS protocol version is now included in open source versions of MatrixSSL. SHA-2 algorithms are now included in the package to support this protocol version.

1.2 ECC Cipher Suites

Elliptic Curve cipher suites are now included. Supported curves are Variations included:

- ECDHE_ECDSA
- ECDH_ECDSA
- ECDHE_RSA
- ECDH_RSA

1.3 Diffie-Hellman Cipher Suites

DH, DHE, and DH_anon cipher suites are now included.

1.4 AES_GCM Cipher Suites

Galois Counter Mode cipher suites are now included for use with TLS 1.2.

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384

1.5 Pre-Shared Key Cipher Suites

PSK cipher suites are now included.

- TLS_DHE_PSK_WITH_AES_256_CBC_SHA
- TLS_DHE_PSK_WITH_AES_128_CBC_SHA
- TLS_PSK_WITH_AES_256_CBC_SHA
- TLS_PSK_WITH_AES_128_CBC_SHA
- TLS_PSK_WITH_AES_256_CBC_SHA384
- TLS_PSK_WITH_AES_128_CBC_SHA256

1.6 ServerNameIndication Hello Extension

Servers can now register a callback to receive the desired hostname that the client sends in the CLIENT_HELLO handshake message. The callback can be used to locate the proper server certificate and key to return to the library to continue the handshake. See the documentation for the new `matrixSslRegisterSNICallback` API.

Clients can now easily include the ServerNameIndication extension in their CLIENT_HELLO message with the new `matrixSslCreateSNIext` helper function.

1.7 Stateless Session Tickets

Support for RFC 5077 is now available. This session resumption mechanism does not require a server-side cache for every session ID. The server issues an encrypted ticket to supporting clients that can then be submitted when resuming.

1.8 Truncated HMAC Extension

Support for the TLS truncated_hmac extension is now included. This feature results in only 10 bytes of HMAC checksum material on each TLS record regardless of the negotiated digest algorithm.

1.9 AES-NI for Supported Linux Intel Processors

AES hardware acceleration for AES-CBC and AES-GCM cipher suites is now included for these platforms.

1.10 SEED and IDEA Symmetric Ciphers

1.11 Client May Choose Subset of Cipher Suites

Previous versions would only allow clients to generate two options for cipher suites in the CLIENT_HELLO; a single cipher suite specified by the caller –OR- every cipher suite that was compiled into the library. Applications may now pass in any number of cipher suites to `matrixSslNewClientSession` that are available for use. This is implemented through a change in the public API for that function. See the API documentation on that function for more information.

1.12 Choose TLS Protocol Version on a Per-Session Basis

Previous versions would only allow clients to generate a CLIENT_HELLO message that specified the highest protocol version that was compiled into the library. Applications may now use the `matrixSslNewClientSession` API to specify the SSL/TLS protocol version to use in the hello message. For example, `USE_TLS_1_2` may be enabled at compile time but a client can now request TLS 1.1 as the highest protocol to support. See the API documentation on that function for more information.

Similarly, servers would always negotiate to a protocol version as long as it had been compiled into the library. The `matrixSslNewServerSession` can now be used to specify the protocol version that it will support when negotiating with a new client. See the API documentation on that function for more information.

1.13 Disable Re-Handshake Support on a Per-Session Basis

Re-handshake support is no longer only a compile-time feature. Peers can disable re-handshaking on a per-session basis using the new `matrixSslDisableRehandshakes` API.

1.14 ZLIB Compression Support

For additional information, see the section in the Developer's Guide.

2 API CHANGES

This is the list of APIs that have changed prototypes or functionality since MatrixSSL 3.4.2. The addition of several new features has introduced many new APIs that have not been available before. For the complete list and discussion of APIs please see the documentation.

2.1 matrixSslNewClientSession

A prototype change has been made to this function. The `cipherSuite` parameter is now an array type and an additional array length parameter has been added to the `matrixSslNewClientSession` API to allow the user to specify a subset of cipher suites that are passed to the user.

This function also now expects a server name in the new `expectedName` parameter to this function. The name is used to test the X.509 certificate received from the server as an extra authentication check.

Also, the final flags parameter to this function can now be used to specify the SSL/TLS protocol version that the client will use when generating the CLIENT_HELLO message.

See the API documentation for details.

2.2 matrixSslNewServerSession

The final flags parameter can now be used to specify the SSL/TLS protocol version that the client will use when generating the CLIENT_HELLO message. See the API documentation for details.

2.3 matrixSslReceivedData

A new return code is possible from this function if using the new zlib compression feature. A code of MATRIXSSL_APP_DATA_COMPRESSED will indicate the application data that has been returned to the caller must be inflated (uncompressed) before use. For more information on the zlib compression feature, see the Developer's Guide.

3 BUG FIXES AND IMPROVEMENTS

3.1 Explicit Length Testing in Parsing Code

An audit of message parsing in MatrixSSL identified a handful of places throughout the code in which explicit tests were needed in areas that read off "length" bytes from message streams. These new tests confirm the values are sane and prevent potential "underflow" decrements of integer counters that would result in bad loop logic. No issues were found that could have been exploited to leak secure information.

3.2 Server Session Cache Improvement

The server-side session cache table to manage client resumption has been greatly improved. The mechanism is now based on a chronological list so the table is no longer 'walked' to find an empty/old entry. The size of the cache table can be much larger without performance degradation as a result of this change.

3.3 Additional X.509 Certificate Authentication Testing

The keyUsage, authorityKeyId, and extendedKeyUsage extensions are now authenticated when processing certificates. The certificate callback now has more documented `alert` and `authStatus` identifiers for the user to see where certificate problems were encountered. The time window for notBefore and notAfter is now done internally in the library (for POSIX platforms). Now only x.509 version 3 certificates are allowed. Certificates without any name identifiers (no commonName or subjectAltName) are rejected.

3.4 Makefile Framework Improvement

The default Makefile system in the product now includes a common.mk configuration file to help detect the host platform and enable compile settings to take advantage of it where applicable.

3.5 Enhanced Example Client and Server Applications

The *./apps/client.c* example client now requires command line parameters to improve the mechanisms of choosing the server IP address, port, session counts, protocol version, and cipher suite. The *runClient.sh* is provided in that directory as an easily editable shell script to launch the client with various parameters.

The *./apps/server.c* example server takes an optional protocol version argument.

3.6 Added 64-bit Div128 For Platforms Missing Internal Capability

Some compilers are missing built-in division operations for 128-bit operators. A software implementation `psDiv128` has been included in *pstm.c* for platforms that have this deficiency.

3.7 NO_RENEGOTIATION Alert Now at Proper Warning Level

This alert used to be incorrectly sent at a fatal level.

3.8 Continued X.509 Certificate Parsing Improvements

Indefinite length ASN.1 encodings are supported. Improvements have been made to DistinguishedName parsing to handle multiple attributes in a single SET.

3.9 Fragmentation Support for CERTIFICATE_REQUEST Message

The max_fragment_length TLS extension is now more fully supported in client authentication handshakes. Previous versions only included support for fragmenting CERTIFICATE messages. The addition of fragmentation support for CERTIFICATE_REQUEST allows large request lists when this extension is active.

3.10 Included the HIGHRES_TIME Code for Testing

The default millisecond granularity in previous versions is often not sufficient for some of the faster crypto tests. A microsecond granularity can be enabled using the `USE_HIGHRES_TIME` define in *./core/osdep.h*