MatrixSSL 3.7.2a

Open Source Release Notes

1 MATRIXSSL 3.7.2B RELEASE NOTES

The 3.7.2b release is a single fix for the 3.7.2(a) version. The patch fixes the AESNI bug to allow AES_GCM cipher suites to decrypt records larger than 4Kb.

2 MATRIXSSL 3.7.2A RELEASE NOTES

The 3.7.2a release is a single fix for the 3.7.2 version. The patch corrects the Elliptic Curve signature format when ECDSA based cipher suites are used.

The ECDSA format is a pair of DER encoded INTEGER values. Previous versions of MatrixSSL were not including a leading 0x0 byte for INTEGER values in which the high bit was set. In strict DER interpretation, a set high bit indicates a negative integer value so TLS implementations that enforce the encoding will reject any INTEGER that appears negative because a negative value is not possible in ECDSA.

The notable TLS implementation that introduced the strict DER testing is OpenSSL 1.0.2.

3 MATRIXSSL 3.7.2 RELEASE NOTES

These release notes highlight the differences between the MatrixSSL 3.7.1 commercial release and this 3.7.2 release.

There are no public API prototype changes in this release.

There are changes to filenames in this release.

3.1 Fixed ALPN extension format for SERVER_HELLO

Servers were incorrectly encoding the Application-Layer Protocol Negotiation reply extension in the SERVER_HELLO handshake message. The example client extension parser has also been fixed to reflect the change.

3.2 Makefile build changes

Optimization levels have defaults based on platform detection in common.mk:

- Assembly language optimizations are enabled if the target CPU supports them: x86_64, i386, ARM, Mips32
- AES-NI instructions are used for Intel processors that support them on Linux and OS X
- Linux, OS X and Windows platforms default to –O3, other platforms to –Os (previously was –Os)
- All objects are built with –ffunction-sections and –fdata-sections, and executables with –Wl,gc-sections to remove any uncalled function from the final binary
- If optimizing for speed (-O[1-3]), larger and faster code is used for MD5, SHA, AES, 3DES, ECC/RSA and 1024 and 2048 bit RSA sizes

Parallel builds (make –j [n]) are now the default, based on the number of cores on the host machine.

MatrixSSL static libraries have been renamed, and dynamic libraries are not longer built, as they were rarely used.

Pre - 3.7.2 Library	3.7.2 Library
libcorestatic.a	libcore_s.a
libpscryptostatic.a	libcrypt_s.a
libmatrixsslstatic.a	libssl_s.a
libcore.[so,dylib,dll]	-
libcrypto.[so,dylib,dll]	-
libmatrixssl.[so,dylib,dll]	-

3.3 Configuration Changes

Several changes were made to the default configuration files (coreConfig.h, cryptoConfig.h and matrixsslConfig.h)

- Enabled by default:
  USE_TLS_RSA_WITH_AES_128_CBC_SHA
  USE_TLS_RSA_WITH_AES_256_CBC_SHA
  USE_TLS_RSA_WITH_AES_128_CBC_SHA256
  USE_TLS_RSA_WITH_AES_256_CBC_SHA256
- USE_SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA no longer enabled by default, and moved to deprecated list
- USE_3DES, USE_PKCS5 and USE_PKCS8 no longer enabled by default.
- BURN_STACK is enabled for all builds, reducing the chance of secret key leakage between functions on the stack

3.4 Improved gcc ARM inline assembly optimizations

Public key math assembly optimizations in pstm_*.c have been fixed to remove compiler warnings (about register constraints) on some ARM tool chains.

3.5 Improved XCode projects to use AES-NI optimizations

XCode projects now include the following default compiler options for AES hardware acceleration:

```
-maes -m64 -mpclmul -msse4.1
```

Note: AES_192 is not supported in AES-NI

Makefile platforms already supported these default options.

Microsoft Windows Visual Studio project files do not currently support these options, although compiling with GCC on Windows will allow AES-NI optimization.

3.6 Improved ECDH key generation

Now ensuring random number is less than the order value when performing ECDH key generation.

3.7 Improved test for absent X.509 subject distingushedName

Previous versions were reporting a certificate was missing a subject dN if the commonName member was missing. Now every dN member must be missing in order for the library to determine there is truly no dN.

3.8 Improved client handling of ServerHello extensions

TLS 1.2 clients will now return the UNSUPPORTED_EXTENSION alert when receiving unexpected (unsolicited) or duplicate extensions from servers. Previous versions would return ILLEGAL_PARAMETER alerts when extension problems were encountered.

3.9 Added more SHA-384 and SHA-512 support

Various signature formats and algorithm parsers were not supporting SHA hashes stronger than SHA-256. Primarily, the additions were to support RSA_PSS variations.

3.10 Improved X.509 support for certs created prior to RFC3280

The KeyUsage extension was not required in old versions of the X.509 specification. If the KeyUsage extension is missing from a CA certificate, a date test is now performed on the notBefore date to see if the certificate was created prior to April 2002. If so, the KeyUsage restriction is bypassed. This is for compatibility with old certificates only, and all newer certificates should have the KeyUsage extension.

3.11 Improved certificate date validation

The notBefore and notAfter dates in X.509 certificates were sometimes being interpreted as being 100 years in the future (and invalid) on very old certificates due to the improper interpretation of 2 digit UTC time.

3.12 Improved stack zeroing

Some compilation platforms would optimize out a memset when the cleared variable was not referenced later in the function. All zeroing memset calls have been changed to a memset_s implementation to force the desired action. OS X and Windows support native API calls for forced memset, but on Linux and other platforms, an implementation is included in core/memset_s.c. The compilation of this file also retains the assembly source output (memset_s.s) for user verification that the memset call has not been optimized out.

This change and other compiler warnings were suggested by Pavel Pimenov using PVS-Studio and Cppcheck. The issues are listed in this blog post and all have been fixed:

http://www.viva64.com/en/b/0304/

3.13 Updated all psFree calls to correct prototype

Several psFree API calls throughout the code base had not been updated to the newer prototype that requires the pool from which the memory was allocated.

3.14 Added URL input to example client

The example client now accepts a `-u` option to provide a URL that will be sent to the host server at the completion of the TLS handshake. This allows the client to request resources from arbitrary HTTPS sites for example:

```
$ nslookup www.google.com
Non-authoritative answer:
Name:     www.google.com
Address: 216.58.216.132
$ ./client -s 216.58.216.132 -p 443 -u /robots.txt
client https://216.58.216.132:443/robots.txt new:1 resumed:0
keylen:1024 nciphers:1 version:TLS 1.2
Using 1024 bit RSA private key
=== 1 new connections ===
...
```

3.15 osdepMutexClose API now called on exit

A typo in psCoreClose prevented osdepMutexClose from being called during library shutdown.

3.16 Corrected buffer length test for ASN.1 SEQUENCE

The function getAsnSequence test for PS_LIMIT_FAIL is now correctly taking into account the length bytes that were read off to retrieve the length. This fix is for a MatrixCMS stream parsing use case and did not affect MatrixSSL users.

3.17 Whitespace cleanup

Source, header and Makefiles were cleaned up by standardizing on linefeeds, removing trailing whitespace, and maximizing tabs on leading whitespace.

3.18 Known issues

- Visual Studio 2013 project files do not properly handle incremental builds. Errors can result in a build (duplicate output objects) unless a “Clean Solution” and “Rebuild Solution” is done each time.
- Windows targets do not support certificate date validation currently. Users requiring this feature can use Windows APIs to get and parse the current date, using the POSIX implementation as a reference.