TLS and Cryptography Libraries Used by MariaDB

Explains how MariaDB links to cryptography libraries (OpenSSL, wolfSSL, GnuTLS, Schannel) either statically or dynamically, and how to verify the active library and version.

When MariaDB Server is compiled with TLS and cryptography support, it is usually either statically linked with MariaDB's bundled TLS and cryptography library or dynamically linked with the system's OpenSSL library. MariaDB's bundled TLS library is either wolfSSL or yaSSL, depending on the server version.

When a MariaDB client or client library is compiled with TLS and cryptography support, it is usually either statically linked with MariaDB's bundled TLS and cryptography library or dynamically linked with the system's TLS and cryptography library, which might be OpenSSL, GnuTLS, or Schannel.

Checking Dynamically vs. Statically Linked

Dynamically linking MariaDB to the system's TLS and cryptography library can often be beneficial, since this allows you to fix bugs in the system's TLS and cryptography library independently of MariaDB. For example, when information on the Heartbleed Bug in OpenSSL was released in 2014, the bug could be mitigated by simply updating your system to use a fixed version of the OpenSSL library, and then restarting the MariaDB Server.

You can verify that `mysqld` is in fact dynamically linked to the OpenSSL shared library on your system by using the ldd command:

```
$ ldd $(which mysqld) | grep -E '(libssl|libcrypto)'
        libssl.so.10 => /lib64/libssl.so.10 (0x00007f8736386000)
        libcrypto.so.10 => /lib64/libcrypto.so.10
(0x00007f8735f25000)
```

If the command does not return any results, then either your `mysqld` is statically linked to the TLS and cryptography library on your system or your `mysqld` is not built with TLS and cryptography support at all.

Checking If the Server Uses OpenSSL

If you aren't sure whether your server is linked with OpenSSL or the bundled TLS library, then you can check the value of the have_openssl system variable. For example:

```
SHOW GLOBAL VARIABLES LIKE 'have_openssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_openssl  | YES   |
+---------------+-------+
```

Checking the Server's OpenSSL Version

If you want to see what version of OpenSSL your server is using, then you can check the value of the version_ssl_library system variable. For example:

```
SHOW GLOBAL VARIABLES LIKE 'version_ssl_library';
+--------------------+---------------------------------+
| Variable_name      | Value                           |
+--------------------+---------------------------------+
| version_ssl_library | OpenSSL 1.0.1e-fips 11 Feb 2013 |
+--------------------+---------------------------------+
```

Note that the version returned by this system variable does not always necessarily correspond to the exact version of the OpenSSL package installed on the system. OpenSSL shared libraries tend to contain interfaces for multiple versions at once to allow for backward compatibility. Therefore, if the OpenSSL package installed on the system is newer than the OpenSSL version that the MariaDB Server binary was built with, then the MariaDB Server binary might use one of the interfaces for an older version. See MDEV-15848 for more information. For example:

```
$ cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.5 (Maipo)
$ rpm -q openssl
openssl-1.0.2k-12.el7.x86_64
$ mysql -u root --batch --execute="SHOW GLOBAL VARIABLES LIKE
'version_ssl_library';"
Variable_name   Value
version_ssl_library     OpenSSL 1.0.1e-fips 11 Feb 2013
$ ldd $(which mysqld) | grep libcrypto
        libcrypto.so.10 => /lib64/libcrypto.so.10
(0x00007f3dd3482000)
$ readelf -a /lib64/libcrypto.so.10 | grep SSLeay_version
  1374: 000000000006f5d0    21 FUNC    GLOBAL DEFAULT   13
SSLeay_version@libcrypto.so.10
  1375: 000000000006f5f0    21 FUNC    GLOBAL DEFAULT   13
SSLeay_version@OPENSSL_1.0.1
  1377: 000000000006f580    70 FUNC    GLOBAL DEFAULT   13
SSLeay_version@@OPENSSL_1.0.2
```

FIPS Certification

Federal Information Processing Standards (FIPS) are standards published by the U.S. federal government that are used to establish requirements for various aspects of computer systems. FIPS 140-2 is a set of standards for security requirements for cryptographic modules.

This standard is relevant when discussing the TLS and cryptography libraries used by MariaDB. Some of these libraries have been certified to meet the standards set by FIPS 140-2.

FIPS Certification by OpenSSL

The OpenSSL library has a special FIPS mode that has been certified to meet the FIPS 140-2 standard. In FIPS mode, only algorithms and key sizes that meet the FIPS 140-2 standard are enabled by the library.

MariaDB does not yet support enabling FIPS mode within the database server. See MDEV-20260 for more information. Therefore, if you would like to use OpenSSL's FIPS mode with MariaDB, then you would either need to enable FIPS mode at the kernel level or enable it via the OpenSSL configuration file, system-wide or only for the MariaDB process.. See the following resources for more information on how to do that:

- Red Hat Enterprise Linux 7: Security Guide: Chapter 8. Federal Standards and Regulations
- Ubuntu Security Certifications Documentation: FIPS for Ubuntu 16.04 and 18.04
- OpenSSL 1.0.2, configuration file method
- OpenSSL 3.0 configuration file method

FIPS Certification by wolfSSL

The standard version of the wolfSSL library has not been certified to meet the FIPS 140-2 standard, but a special "FIPS-ready" version has been certified. Unfortunately, the "FIPS-ready" version of wolfSSL uses a license that is incompatible with MariaDB's license, so it cannot be used with MariaDB.

FIPS Certification by yaSSL

The yaSSL library has not been certified to meet the FIPS 140-2 standard.

Libraries Used by Each Platform and Package

MariaDB Server

MariaDB Server on Windows

MariaDB Server is statically linked with the bundled wolfSSL ↗ library in MSI and ZIP packages on Windows.

MariaDB Server on Linux

MariaDB Server in Binary Tarballs

MariaDB Server is statically linked with the bundled wolfSSL ↗ library in binary tarballs on Linux.

MariaDB Server in DEB Packages

MariaDB Server is dynamically linked with the system's OpenSSL ↗ library in .deb ↗ packages.

See Differences in MariaDB in Debian (and Ubuntu) for more information.

MariaDB Server in RPM Packages

MariaDB Server is dynamically linked with the system's OpenSSL ↗ library in .rpm ↗ packages.

MariaDB Clients and Utilities

MariaDB Connector/C has been included with MariaDB Server, and the bundled and the clients and utilities are linked with it. On some platforms, MariaDB Connector/C and these clients and utilities ↗ may use a different TLS library than the one used by MariaDB Server and libmysqlclient ↗.

MariaDB Clients and Utilities on Windows

MariaDB's clients and utilities and MariaDB Connector/C are dynamically linked with the system's Schannel libraries in MSI and ZIP packages on Windows. libmysqlclient is still statically linked with the bundled wolfSSL library.
<>

MariaDB Clients and Utilities on Linux

MariaDB Clients and Utilities in Binary Tarballs

MariaDB's clients and utilities and MariaDB Connector/C are statically linked with the GnuTLS library in binary tarballs on Linux. libmysqlclient is still statically linked with the bundled wolfSSL library.
<>

MariaDB Clients and Utilities in DEB Packages

MariaDB's clients and utilities, libmysqlclient, and MariaDB Connector/C are dynamically linked with the system's OpenSSL library in .deb packages.

See Differences in MariaDB in Debian (and Ubuntu) for more information.

MariaDB Clients and Utilities in RPM Packages

MariaDB's clients and utilities, libmysqlclient, and MariaDB Connector/C are dynamically linked with the system's OpenSSL library in .rpm packages.

Updating Dynamically Linked OpenSSL Libraries on Linux

When the MariaDB Server or clients and utilities are dynamically linked to the system's OpenSSL library, it makes it very easy to update the libraries. The information below will show how to update these libraries for each platform.

Updating Dynamically Linked OpenSSL Libraries with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to update the libraries using yum ↗ or dnf ↗. Starting with RHEL 8 and Fedora 22, `yum` has been replaced by `dnf`, which is the next major version of `yum`. However, `yum` commands still work on many systems that use `dnf`. For example:

Update the package by executing the following command:

```
sudo yum update openssl
```

And then restart MariaDB server and any clients or applications that use the library.

Updating Dynamically Linked OpenSSL Libraries with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to recommended to update the libraries using apt-get ↗. For example:

First update the package cache by executing the following command:

```
sudo apt update
```

And then update the package by executing the following command:

```
sudo apt-get update openssl
```

And then restart ↗ MariaDB server and any clients or applications that use the library.

Updating Dynamically Linked OpenSSL Libraries with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to recommended to update the libraries using zypper ↗. For example:

Update the package by executing the following command:

```
sudo zypper update openssl
```

And then restart MariaDB server and any clients or applications that use the library.

Previous
Encryption

Next
Data-in-Transit Encryption

Last updated 14 days ago

Was this helpful?