Securing Connections for Client and Server

Complete MariaDB security guide. Complete resource for user management, access control, SSL/TLS encryption, and audit policies with comprehensive examples.

Starting from 11.4 MariaDB, it encrypts the transmitted data between the server and clients by default unless the server and client run on the same host.

Before that the default behavior was to transmit the data unencrypted over the network introducing a security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

The data in transit are encrypted (by default or if enabled manually) using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix `ssl_`, but internally, MariaDB only supports its secure successors.

Enabling TLS

Enabling TLS for MariaDB Server

Current < 11.4

MariaDB enables TLS automatically. Certificates are generated on startup and only stored in memory. Certificate verification is enabled by default on the client side and certificates are verified if the authentication plugin itself is MitM safe (`mysql_native_password` , `ed25519` , `parsec`).

Reloading the Server's Certificates and Keys Dynamically

The `FLUSH SSL` command can be used to dynamically reinitialize the server's TLS context.

See FLUSH SSL for more information.

Enabling TLS for MariaDB Clients

Different clients and utilities may use different methods to enable TLS. You can let the client to use TLS without specifying client-side certificate — this is called a **one-way TLS** below — to have the connection encrypted. Or you can additionally provide client-side certificate — this is **two-way TLS** — which will allow the server to do the certificate based client authentication.

Enabling One-Way TLS for MariaDB Clients

One-way TLS means that only the server provides a private key and an X509 certificate. When TLS is used without a client certificate, it is called "one-way" TLS, because only the server can be authenticated, so certificate based authentication is only possible in one direction. However, encryption is still possible in both directions. Server certificate verification means that the client verifies that the certificate belongs to the server.

MariaDB starting with 11.4

Starting from MariaDB 11.4 (Connector/C version 3.4) this mode is enabled by default. Connector/C will enable TLS automatically on all non-local connections and will require a verified server certificate to prevent man-in-the-middle attacks.

To enable one-way TLS manually (for older clients or if you want verification with CA certificate) you can specify these options in a relevant client option group in an option file:

```
[client-mariadb]
...
ssl_ca = /etc/my.cnf.d/certificates/ca.pem
ssl-verify-server-cert
```

Or if you wanted to specify them on the command-line with the mariadb client, then you could execute something like this:

```
$ mariadb -u myuser -p -h myserver.mydomain.com \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --ssl-verify-server-cert
```

You can disable server certificate verification with

```
[client-mariadb]
disable-ssl-verify-server-cert
```

(or a similar command line option), but it creates a risk of a man-in-the-middle attack where the connection can be eavesdropped or manipulated even if it is being encrypted.

Enabling Two-Way TLS for MariaDB Clients

Two-way TLS means that both the client and server provide a private key and an X509 certificate. It is called "two-way" TLS because both the client and server can be authenticated.

For example, to specify these options in a relevant client option group in an option file, you could set the following:

```
[client-mariadb]
...
ssl_cert = /etc/my.cnf.d/certificates/client-cert.pem
ssl_key = /etc/my.cnf.d/certificates/client-key.pem
ssl_ca = /etc/my.cnf.d/certificates/ca.pem
ssl-verify-server-cert
```

Or if you wanted to specify them on the command-line with the mariadb client, then you could execute something like this:

```
$ mariadb -u myuser -p -h myserver.mydomain.com \
   --ssl-cert=/etc/my.cnf.d/certificates/client-cert.pem \
   --ssl-key=/etc/my.cnf.d/certificates/client-key.pem \
   --ssl-ca=/etc/my.cnf.d/certificates/ca.pem \
   --ssl-verify-server-cert
```

Two-way SSL is required for an account if the `REQUIRE X509`, `REQUIRE SUBJECT`, and/or `REQUIRE ISSUER` clauses are specified for the account.

Enabling TLS for MariaDB Connector/C Clients

See the documentation on MariaDB Connector/C's TLS Options for information on how to enable TLS for clients that use MariaDB Connector/C.

Enabling TLS for MariaDB Connector/ODBC Clients

See the documentation on MariaDB Connector/ODBC's TLS-Related Connection Parameters for information on how to enable TLS for clients that use MariaDB Connector/ODBC.

Enabling TLS for MariaDB Connector/J Clients

See the documentation on Using TLS/SSL with MariaDB Connector/J for information on how to enable TLS for clients that use MariaDB Connector/J.

Verifying that a Connection is Using TLS

You can verify that a connection is using TLS by checking the connection's Ssl_version status variable. It will either show the TLS version used, or be empty when no encryption is in effect. For example:

```
SHOW SESSION STATUS LIKE 'Ssl_version';
+---------------+---------------------------+
| Variable_name | Value                     |
+---------------+---------------------------+
| Ssl_version   | TLSv1.3                   |
+---------------+---------------------------+
1 row in set (0.00 sec)
```

Requiring TLS

From MariaDB 10.5.2, the require_secure_transport system variable is available. When set (by default it is off), connections attempted using insecure transport will be rejected. Secure transports are SSL/TLS, Unix sockets or named pipes. Note that requirements set for specific user accounts will take precedence over this setting.

Requiring TLS for Specific User Accounts

You can set certain TLS-related restrictions for specific user accounts. For instance, you might use this with user accounts that require access to sensitive data while sending it across networks that you do not control. These restrictions can be enabled for a user account with the CREATE USER, ALTER USER, or GRANT statements. For example:

- A user account must connect via TLS if the user account is defined with the `REQUIRE SSL` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE SSL;
```

- A user account must connect via TLS with a specific cipher if the user account is defined with the `REQUIRE CIPHER` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE CIPHER 'ECDH-RSA-AES256-SHA384';
```

- A user account must connect via TLS with a valid client certificate if the user account is defined with the `REQUIRE X509` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE X509;
```

- A user account must connect via TLS with a specific client certificate if the user account is defined with the `REQUIRE SUBJECT` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/
L=Portland';
```

- A user account must connect via TLS with a client certificate that must be signed by a specific certificate authority if the user account is defined with the `REQUIRE ISSUER` clause.

```
ALTER USER 'alice'@'%'
   REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/
L=Portland'
   AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter
Parker/emailAddress=p.parker@marvel.com';
```

Requiring TLS for Specific User Accounts from Specific Hosts

A user account can have different definitions depending on what host the user account is logging in from. Therefore, it is possible to have different TLS requirements for the same username for different hosts. For example:

```
CREATE USER 'alice'@'localhost'
   REQUIRE NONE;

CREATE USER 'alice'@'%'
   REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/
L=Portland'
   AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter
Parker/emailAddress=p.parker@marvel.com'
   AND CIPHER 'ECDHE-ECDSA-AES256-SHA384';
```

In the above example, the `alice` user account does not require TLS when logging in from localhost. However, when the `alice` user account logs in from any other host, they must use TLS with the given cipher, and they must provide a valid client certificate with the given subject that must have been signed by the given issuer.

Previous
Secure Connections Overview

Next
SSL/TLS System Variables