**Securing the MySQL database and Java Code**

Reference:<https://dev.mysql.com/doc/refman/5.7/en/security.html>

**Password Policy**:

Based on guidance from the National Institute of Standards and Technology (NIST) passwords should:

· Be a minimum of 8 characters long

· Not use dictionary words

· Not use repetitive or successive characters (Ex: ‘aaaaaaaa’, ‘1234abcd’)

· Avoid context specific words (names, work roles, generic default passwords (ie: ‘password’) Note: should follow this guidance for usernames as well.

· Should be checked against a blacklist of known compromised or bad passwords

· Be stored on the system securely (using hashing)

· Use rate throttling to slow down repeated authentication failures and eventually lock out the account.

· Password and/or connection it is being transferred over should be encrypted.

The NIST guidance also goes on to say how complexity of passwords is less important as it seems to result in frustration for users and users not following best practices for password creation. Instead they suggest the validator be more thorough to follow all the above guidance

Resource: NIST Digital Identity Guidelines (Sections 5.1.1.1, 5.1.1.2, and Appendix A)<https://pages.nist.gov/800-63-3/sp800-63b.html>

**Secure Connections:**

<https://dev.mysql.com/doc/refman/5.7/en/secure-connections.html>

MySQL Community Edition source distributions can be compiled using either OpenSSL or yaSSL (see [Section 7.4.2, “Building MySQL with Support for Secure Connections”](https://dev.mysql.com/doc/refman/5.7/en/building-with-secure-connection-support.html)).

<http://www.openssl.org>.

Uses TLS connections:<https://dev.mysql.com/doc/refman/5.7/en/secure-connection-protocols-ciphers.html>

· Only allow TLS version 1.2 in the server my.cnf file.

· For better security, use a certificate with an RSA key size of 2048 bits or more, ideally should use a registered Certificate Authority but in our case we’ll just use self-signed Certs.

· SSL files (certs) generated are valid for 10 years, RSA keys do not expire.

**Security Plug-ins:**

Security plug-ins for MySQL that are of interest for this project are below:

**Secure Password Storage:**

SHA-256 Plugin

<https://dev.mysql.com/doc/mysql-security-excerpt/5.6/en/sha256-authentication-plugin.html>

**Failed authentication attempt slowdown/lockout:**

Connection-Control Plugin

<https://dev.mysql.com/doc/refman/5.7/en/connection-control-plugin.html>

**Password Policy Control**

Password Validation Plugin

<https://dev.mysql.com/doc/refman/5.7/en/validate-password-plugin.html>

**Principle of least privilege:**

Database roles (admin, supervisor, user)

The idea here is that each role will only have the privileges required to accomplish their task and nothing more. So a user can only see and modify their data, a supervisor can see users under them and modify their data for them (we should make sure to log all this), and an admin can pretty much change anything (this access should be limited to only a few admins who require access, and should use usernames that uniquely identify them for accounting purposes).

**Security of installation:**

Protection of log files, data files, and application files on the system.

**Backups and Recovery:**

For database, configuration, and log files.

<https://dev.mysql.com/doc/refman/5.7/en/backup-and-recovery.html>

**Network Security guidance:**

Allowing remote MySQL only to systems that require it for operational purpose. Host based firewalls should restrict access to only those systems that are required for the service to work, ideally network based firewalls or access control lists (ACLs) will also be in place to prevent unnecessary and possibly malicious connections as well.

**Application security (Java Code)**:

SQL injection (SQLi) is an attack where a malicious user can execute SQL statements that can retrieve or possibly even modify data on the database server, in some instances they are able to do this without requiring any authentication. The vulnerability exists if a web application allows direct user input to be “injected” into a SQL statement as is often the case for authentication requests. It is imperative to prevent these types of attacks on your system to avoid data leakage or corruption by a malicious user. In addition to following least privileged access as noted above, the input that is taken on the web application must be validated to ensure only the expected data is entered.

Java - use PreparedStatement() with bind variables

See:<https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet>

**Additional Notes:**

**Configuring Sever:**

·  [--ssl-ca](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl-ca) identifies the Certificate Authority (CA) certificate.

·  [--ssl-cert](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl-cert) identifies the server public key certificate. This can be sent to the client and authenticated against the CA certificate that it has.

·  [--ssl-key](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl-key) identifies the server private key.

For example, start the server with these lines in the my.cnf file, changing the file names as necessary:

[mysqld]

ssl-ca=ca.pem

ssl-cert=server-cert.pem

ssl-key=server-key.pem

[require\_secure\_transport](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_require_secure_transport) SEE: [Section 6.1.5, “Server System Variables”](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html)

**Configuring Client:**

For client programs, options for secure connections are similar to those used on the server side, but [--ssl-cert](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl-cert) and [--ssl-key](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl-key) identify the client public and private key:

·  [--ssl-ca](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl-ca) identifies the Certificate Authority (CA) certificate. This option, if used, must specify the same certificate used by the server.

·  [--ssl-cert](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl-cert) identifies the client public key certificate.

·  [--ssl-key](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl-key) identifies the client private key.

To connect securely to a MySQL server that supports secure connections, the options that a client must specify depend on the encryption requirements of the MySQL account used by the client. (See the discussion of the REQUIRE clause in [Section 14.7.1.2, “CREATE USER Syntax”](https://dev.mysql.com/doc/refman/5.7/en/create-user.html).)

To require that a client certificate also be specified, create the account using the REQUIRE X509 option

To require a secure connection and fail if one cannot be established, invoke the client with [--ssl-mode=REQUIRED](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl-mode), [--ssl](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl), or a synonym ([--ssl=1](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl), [--enable-ssl](https://dev.mysql.com/doc/refman/5.7/en/secure-connection-options.html#option_general_ssl)).