Configuring SSL for FIPS 140-2 compatibility

Enterprise Edition Introduced in 5.24

Federal Information Processing Standards (FIPS) 140 is a U.S. government standard established by the National Institute of Standards and Technology (NIST) which is used to accredit cryptographic modules such as those used in TLS network encryption. While FIPS 140 compliance is primarily required for federal agencies and their contractors, it also is used in the healthcare sector under regulations like the Health Insurance Portability and Accountability Act (HIPAA) to protect patient data.

This guide helps configure Neo4j to use TLS/SSL encryption in a FIPS-compliant way. It is supplementary to the SSL framework documentation, as many of the configuration processes and requirements are the same.

Prerequisites

- Verify that the machine running Neo4j has FIPS-compatible hardware and operating system. Only Linux operating systems are supported for Neo4j FIPS compatibility at this time.
- Use Neo4j Enterprise 5.23.0 or later.
- Install and configure a non-native authentication provider, for example LDAP or SSO. See Authentication and authorization.

Enable FIPS SSL provider

The secure networking in Neo4j is provided through the Netty library, which supports both the native JDK SSL provider and Netty-supported OpenSSL derivatives. Specifically Netty's *Forked Tomcat Native* library called netty-tcnative→ .

The `netty-tcnative` library is provided in several variants. However, to achieve AI search
you must use the dynamically linked version of `netty-tcnative` alongside a FIPS-compatible

installation of OpenSSL.

The dynamically linked library requires the following dependencies to be installed[1]:

- Apache Portable Runtime Library
- A FIPS certified version of OpenSSL, with a FIPS provider installed and set as default.

Refer to Forked Tomcat Native → for more information.

NOTE

Netty provides a convenient pre-build, statically linked version of `netty-tcnative` using BoringSSL, but this is not FIPS certified[2].

By using the dynamic `netty-tcnative` library variant combined with a FIPS certified OpenSSL installation, Neo4j's cryptographic operations are delegated by `netty-tcnative` to OpenSSL, transitively giving FIPS compatibility.

Install Apache portable runtime library

To install Apache Portable Runtime Library →, use the operating system's package manager.

In Debian/Ubuntu this package is usually called `libapr1`

Install Apache Portable Runtime Library in Debian or Ubuntu

```
apt install -y libapr1
```

In RedHat Enterprise Linux, the package is usually called `apr`:

Install Apache Portable Runtime Library in RedHat

```
dnf install -y apr
```

Install OpenSSL

Instructions on how to build and install a FIPS-compatible OpenSSL are out of scope for this document. Installation steps can differ depending on operating system, and other security requirements you might have for OpenSSL.

In general:

- For a list of FIPS certified OpenSSL versions, see https://openssl-library.org/source/ → .
- A FIPS provider must be installed into OpenSSL.
- OpenSSL must be configured to use the FIPS provider by default.

Install the correct `netty-tcnative` library

Builds of `netty-tcnative` dynamic library are provided in the Neo4j `lib` directory under their own subfolder called `netty-tcnative`.

To install the `netty-tcnative` dynamic library:

1. Locate the Neo4j `lib` directory.

   The location of the `lib` directory is different depending on the method used to install Neo4j. Check the file locations documentation for the correct location.

   This location will be referred to as *<NEO4J_LIB>*.

2. Make sure there are no `netty-tcnative-boringssl` libraries present in the *<NEO4J_LIB>* folder.

   ```
   find <NEO4J_LIB> -name "netty-tcnative-boringssl*.jar" -delete
   ```

3. Check which netty-tcnative libraries are available:

   ```
   ls -l <NEO4J_LIB>/netty-tcnative
   ```

   There are Linux and Fedora Linux variants available, compiled for both x86_64 and ARM 64 architectures. Select the one matching the local machine's operating system and architecture.

4. Verify the dependencies are correctly installed using `ldd` →:

Verify netty-tcnative dependencies are installed

```
unzip -d /tmp <NEO4J_LIB>/netty-tcnative/netty-tcnative-*-linux-$(arch).jar
ldd /tmp/META-INF/native/libnetty_tcnative_linux_*.so
rm -rf /tmp/META-INF
```

Verify Fedora variant of netty-tcnative dependencies are installed

```
unzip -d /tmp <NEO4J_LIB>/netty-tcnative/netty-tcnative-*-linux-$(arch)-fedora.jar
ldd /tmp/META-INF/native/libnetty_tcnative_linux_$(arch).so
rm -rf /tmp/META-INF
```

The `ldd` command shows a list of library dependencies and where they are loaded from on the local machine.

- If any dependencies are missing, they must be installed, or Neo4j will fail to run.
- The `libssl.so` and `libcrypto.so` libraries listed must be the ones installed with OpenSSL in the previous steps.

5. Copy the verified JAR file to *<NEO4J_LIB>*.

NOTE

Only copy **one** of the JAR files. Otherwise Neo4j will not be able to resolve dependencies at runtime. In case of this error, you will get a message like:

```
"Failed to load any of the given libraries: [netty_tcnative_linux_x86_64,
netty_tcnative_linux_x86_64_fedora, netty_tcnative_x86_64, netty_tcnative]".
```

Generate SSL certificate and private key

Neo4j SSL encryption requires a certificate in the X.509 standard and a private key in PKCS #8 format, both encoded in PEM format.

IMPORTANT

IMPORTANT

For FIPS compatibility, the private key must be secured with a password.

Refer to the SSL certificate and key instructions for more information.

Configure Neo4j to use SSL encryption

SSL configuration is described in detail in SSL framework configuration.

This section describes configuration that must be done **in addition to** standard non-FIPS compliant SSL configuration.

NOTE

- The following group of FIPS-compatible cipher suites is for use with TLSv1.2:
  - `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
  - `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384`
  - `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`

  They require additional configuration in the application or OpenSSL settings.

- The following cipher suites are supported by default in OpenSSL when using TLSv1.3:
  - `TLS_AES_256_GCM_SHA384`
  - `TLS_AES_128_GCM_SHA256`

  These suites do not require additional configuration when OpenSSL is built with FIPS support.

Bolt

1. Set `dbms.netty.ssl.provider=OPENSSL`
2. Set `server.bolt.tls_level=REQUIRED`
3. Follow instructions on how to Configure SSL over Bolt.
4. Set additional Bolt configurations:

```
dbms.ssl.policy.bolt.trust_all=false
dbms.ssl.policy.bolt.tls_level=REQUIRED
dbms.ssl.policy.bolt.tls_versions=TLSv1.2,TLSv1.3
dbms.ssl.policy.bolt.ciphers=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_12
8_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_AES
_256_GCM_SHA384,TLS_AES_128_GCM_SHA256
```

5. Follow the instructions in SSL Framework → Using encrypted private key to configure `dbms.ssl.policy.bolt.private_key_password` to dynamically read the password from an encrypted password file. The password must **not** be set in plain text.

HTTPS

This section is only applicable if HTTPS is enabled.

1. Follow instructions on how to Configure SSL over HTTPS.
2. Set additional HTTPS configurations:

```
dbms.ssl.policy.https.trust_all=false
dbms.ssl.policy.https.tls_level=REQUIRED
dbms.ssl.policy.https.tls_versions=TLSv1.2,TLSv1.3
dbms.ssl.policy.https.ciphers=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_1
28_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_AE
S_256_GCM_SHA384,TLS_AES_128_GCM_SHA256
```

3. Follow the instructions in SSL Framework → Using encrypted private key to configure `dbms.ssl.policy.https.private_key_password` to dynamically read the password from an encrypted password file. The password must NOT be set in plain text.

Intra-cluster encryption

For FIPS compatbility, intra-cluster encryption must be enabled if you are running a Neo4j cluster.

1. Follow instructions to configure SSL for intra-cluster communication.
2. Set additional cluster configurations:

```
dbms.ssl.policy.cluster.enabled=true
dbms.ssl.policy.cluster.tls_level=REQUIRED
dbms.ssl.policy.cluster.client_auth=REQUIRED
dbms.ssl.policy.cluster.tls_versions=TLSv1.2,TLSv1.3
dbms.ssl.policy.cluster.ciphers=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES
_128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_
AES_256_GCM_SHA384,TLS_AES_128_GCM_SHA256
```

3. Follow the instructions in SSL Framework → Using encrypted private key to configure `dbms.ssl.policy.cluster.private_key_password` to dynamically read the password from an encrypted password file. The password must **not** be set in plain text.

Backup

This section is applicable on instances or cluster members used for taking backups.

1. Follow instructions on how to Configure SSL for backup communication.
2. Set additional backup configurations:

```
dbms.ssl.policy.backup.enabled=true
dbms.ssl.policy.backup.client_auth=REQUIRED
dbms.ssl.policy.backup.trust_all=false
dbms.ssl.policy.backup.tls_versions=TLSv1.2,TLSv1.3
dbms.ssl.policy.backup.ciphers=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_
128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_A
ES_256_GCM_SHA384,TLS_AES_128_GCM_SHA256
```

3. Follow the instructions in SSL Framework → Using encrypted private key to configure `dbms.ssl.policy.backup.private_key_password` to dynamically read the password from an encrypted password file. The password must **not** be set in plain text.

---

1. https://netty.io/wiki/forked-tomcat-native.html →

2. https://boringssl.googlesource.com/boringssl/+/master/crypto/fipsmodule/FIPS.md →

Prev
‹ SSL framework

Next
Browser credentials handling ›

Contents

Neo4j Videos

GraphAcademy

Neo4j Labs

Want to Speak?

France: +33 (0) 1 88 46 13 20