

Neo4j Customer Success:

SSL Configuration

Ali Maddahian

Customer Success Architect

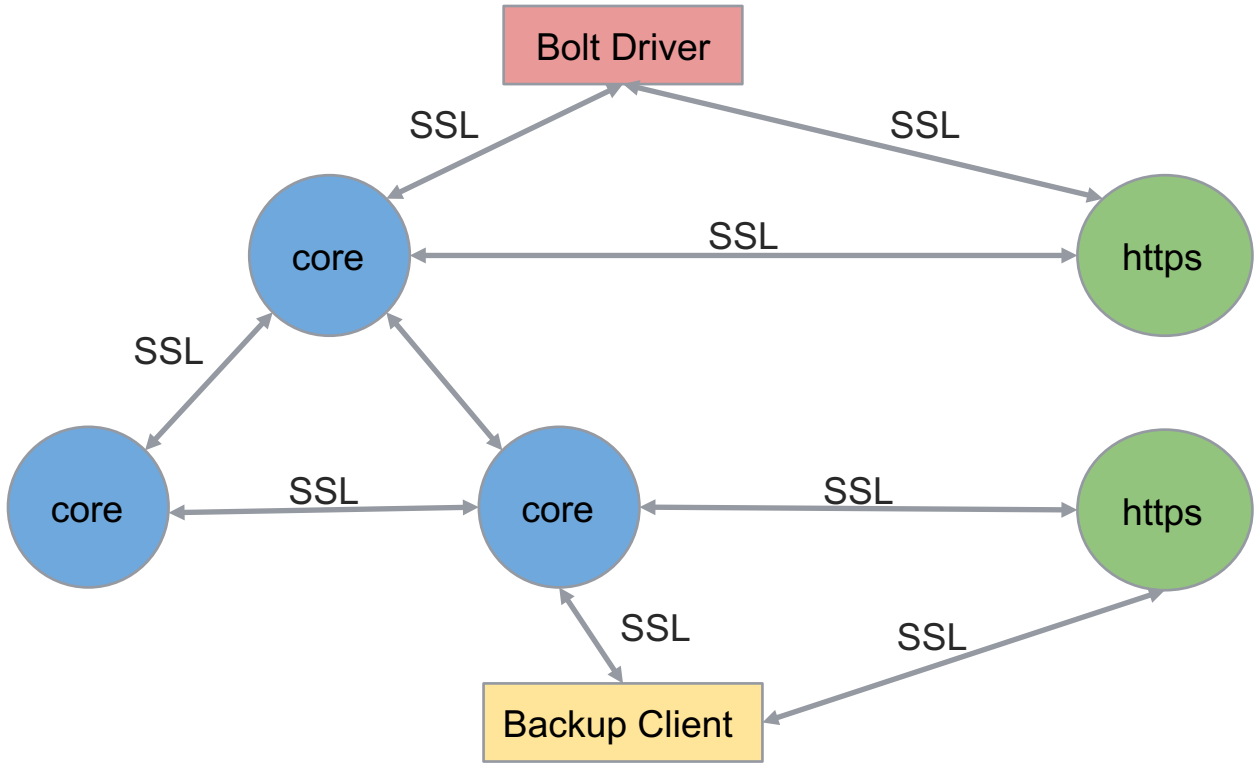
neo4j

Neo4j Customer Success
SSL Configuration

- SSL Framework Overview
- Certificate Generation
- Neo4j SSL Configuration
    - https
    - bolt
    - Intra-Cluster
    - Backup
- Validation

SSL Framework Overview

Secure cluster, backup and client



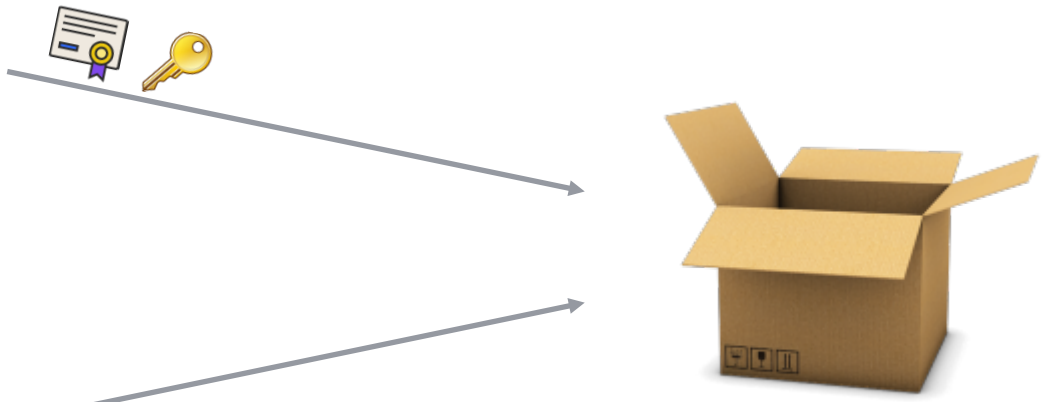
framework for SSL

Cryptographic objects

```
private key
public certificate
trusted certificates
revoked certificates
```

Cryptographic parameters

```
ciphers
tls protocol versions
mutual authentication
```



A boxed SSL policy

File system structure

certificates/<policyname>
certificates/<policyname>/private.key
certificates/<policyname>/public.crt
certificates/<policyname>/trusted/
certificates/<policyname>/revoked/

relative to $NEO4J_HOME
override with absolute paths

Example cluster policy

```
dbms.ssl.policy.cluster.base_directory = certificates/cluster

# These are all defaults
#dbms.ssl.policy.cluster.private_key = private.key
#dbms.ssl.policy.cluster.public_certificate = public.crt
#dbms.ssl.policy.cluster.trusted_dir = trusted
#dbms.ssl.policy.cluster.revoked_dir = revoked

dbms.ssl.policy.cluster.client_auth = REQUIRE
dbms.ssl.policy.cluster.tls_versions = TLSv1.2
dbms.ssl.policy.cluster.ciphers = TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

# Apply the cluster policy
causal_clustering.ssl_policy = cluster
```

Certificate Generation

Install Software

Launch an EC2 Ubuntu 18 instance

Assign an Elastic IP address to your EC2 instance

Create a domain name for your IP

Example(free): https://freedns.afraid.org/subdomain/

Install Java

```
sudo apt-get update
sudo apt install openjdk-8-jdk openjdk-8-jre
# For 4.0 - sudo apt install openjdk-11-jdk openjdk-11-jre
```

Install Neo4j

```
tar xvzf neo4j-enterprise-3.5.17-unix.tar.gz
# tar xvzf neo4j-enterprise-4.0.4-unix.tar.gz
```

Install Let'sEncrypt

```
sudo apt-get update
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get install -y certbot
```

Generate CA Signed Certificates (*LetsEncrypt*)

Generate signed public certificate and private key

$sudo certbot certonly
$sudo chgrp -R ubuntu /etc/letsencrypt/*
$sudo chmod -R g+rx /etc/letsencrypt/*

Provide your email address, and domain name,
and agree to licensing for certificate generation.
should your domain name changes later, you can

delete the current certificate by issuing
$sudo certbot delete
and then regenerate a new one using **certonly** option

IMPORTANT NOTES:

- Congratulations! Your certificate and chain have been saved at:
/etc/letsencrypt/live/MY_DOMAIN/fullchain.pem
Your key file has been saved at:
/etc/letsencrypt/live/MY_DOMAIN/privkey.pem
*Your cert will expire on 2020-08-02. To obtain a new or tweaked version of this certificate in the future, simply run certbot again. To non-interactively renew *all* of your certificates, run "certbot renew"*

- Your account credentials have been saved in your Certbot configuration directory at /etc/letsencrypt. You should make a secure backup of this folder now. This configuration directory will also contain certificates and private keys obtained by Certbot so making regular backups of this folder is ideal.

Self-Signed Certificates

3.5.x

- Controlled by *(i.e. dbms.ssl.policy.client_policy.allow_key_generation=true)*
- *Assuming we a policy called "client_policy"*
- *Generated upon restart and placed in the certificates directory*
- *Not generally recommended*
- ***Removed in 4.x***

Alternatively, you can create one manually via openssl as an example

- openssl req -x509 -newkey rsa:2048 -keyout private_key.pem -out public_cert.pem -days 10

Neo4j SSL Configuration

Neo4j (3.5.x) Cluster Configuration on AWS - Example

- causal_clustering.minimum_core_cluster_size_at_formation=3
- causal_clustering.minimum_core_cluster_size_at_runtime=3
- causal_clustering.initial_discovery_members=10.0.0.55:5000,10.0.0.60:5000,10.0.0.53:5000
- causal_clustering.raft_advertised_address=10.0.0.53:7000
- causal_clustering.transaction_advertised_address=10.0.0.53:6000

Neo4j (3.5.x) SSL Configuration – https/bolt

Configure Neo4j for SSL (https/bolt connector SSL policy named client_policy in this example)

dbms.connector.bolt.enabled=true

dbms.connector.https.enabled=true

dbms.connector.bolt.advertised_address=mydomain.com:7687

dbms.connector.https.advertised_address=mydomain.com:7473

bolt.ssl_policy=client_policy

https.ssl_policy=client_policy

dbms.ssl.policy.client_policy.allow_key_generation=false

dbms.ssl.policy.client_policy.client_auth=NONE

dbms.ssl.policy.client_policy.base_directory =/home/ubuntu/neo4j-enterprise-3.5.17/certificates/client_policy

dbms.ssl.policy.client_policy.private_key = /home/ubuntu/neo4j-enterprise-3.5.17/certificates/client_policy/private.key

dbms.ssl.policy.client_policy.public_certificate =/home/ubuntu/neo4j-enterprise-3.5.17/certificates/client_policy/public.cert

dbms.ssl.policy.client_policy.revoked_dir =/home/ubuntu/neo4j-enterprise-3.5.17/certificates/client_policy/revoked

dbms.ssl.policy.client_policy.trusted_dir =/home/ubuntu/neo4j-enterprise-3.5.17/certificates/client_policy/trusted

Neo4j SSL Configuration – https/bolt

```
$cd neo4j-enterprise-3.5.17/certificates/
$mkdir  client_policy
$cd client_policy
$mkdir trusted
$mkdir revoked

# Assuming using Let'sEncrypt
$ln -s /etc/letsencrypt/live/$MY_DOMAIN/privkey.pem        private.key
$ln -s /etc/letsencrypt/live/$MY_DOMAIN/fullchain.pem      public.crt
$ln -s /etc/letsencrypt/live/$MY_DOMAIN/fullchain.pem      trusted/public.crt

$./neo4j restart
```

Neo4j SSL Configuration – https/bolt

- The HTTPS and Bolt servers do not support client authentication (a.k.a. mutual authentication). As a result, client_auth is set to be turned off explicitly by having client_auth=NONE
- The tls_versions and ciphers settings are supported
- Both public certificate and private key files need to be in PEM format.
  - Private key - **PKCS#8** key in PEM format.
  - Certificate - **X.509** certificate (chain) of the server in PEM

Specific Cyphers

- You can specific specific ciphers to be used for your configuration:
- Example:
  - dbms.ssl.policy.test.ciphers=TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

- debug.log output
  - 2020-04-07 00:08:15.657+0000 WARN Weak cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA enabled for SslContextFactory@78fb72cb[provider=null,keyStore=null,trustStore=null]

Intra-Cluster SSL Policy(3.5.x)

causal_clustering.ssl_policy=cluster_policy

dbms.ssl.policy.cluster_policy.ciphers=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

dbms.ssl.policy.cluster_policy.allow_key_generation=false

dbms.ssl.policy.cluster_policy.client_auth=REQUIRE

dbms.ssl.policy.cluster_policy.base_directory=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/cluster_policy

dbms.ssl.policy.cluster_policy.private_key=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/cluster_policy/private.key

dbms.ssl.policy.cluster_policy.public_certificate=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/cluster_policy/public.crt

dbms.ssl.policy.cluster_policy.revoked_dir=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/cluster_policy/revoked

dbms.ssl.policy.cluster_policy.trusted_dir=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/cluster_policy/trusted

Backup SSL Policy (3.5.x)

dbms.backup.ssl_policy=backup_policy

dbms.ssl.policy.backup_policy.ciphers=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

dbms.ssl.policy.backup_policy.allow_key_generation=false

dbms.ssl.policy.backup_policy.client_auth=REQUIRE

dbms.ssl.policy.backup_policy.base_directory=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/backup_policy

dbms.ssl.policy.backup_policy.private_key=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/backup_policy/private.key

dbms.ssl.policy.backup_policy.public_certificate=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/backup_policy/public.crt

dbms.ssl.policy.backup_policy.revoked_dir=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/backup_policy/revoked

dbms.ssl.policy.backup_policy.trusted_dir=/home/ubuntu/neo4j-enterprise-3.5.17/certificates/backup_policy/trusted

Certificate Directory Listing (3.5.x)

```
$cd $NEO4J_HOME/certificates
$find . -print | xargs ls -l

./backup_policy/public.crt -> /etc/letsencrypt/live/cc3-2.my.to/fullchain.pem
./backup_policy/private.key -> /etc/letsencrypt/live/cc3-2.my.to/privkey.pem
./backup_policy/trusted/public.crt -> /etc/letsencrypt/live/cc3-2.my.to/fullchain.pem

./client_policy/public.crt -> /etc/letsencrypt/live/cc3-2.my.to/fullchain.pem
./client_policy/private.key -> /etc/letsencrypt/live/cc3-2.my.to/privkey.pem
./client_policy/trusted/public.crt -> /etc/letsencrypt/live/cc3-2.my.to/fullchain.pem

./cluster_policy/public.crt -> /etc/letsencrypt/live/cc3-2.my.to/fullchain.pem
./cluster_policy/private.key -> /etc/letsencrypt/live/cc3-2.my.to/privkey.pem
./cluster_policy/trusted/public.crt -> /etc/letsencrypt/live/cc3-2.my.to/fullchain.pem
```

* MY_DOMAIN is set to cc3-2.my.to

Neo4j 4.0 Configuration (neo4j.conf)

<scope> = [bolt | https | cluster | backup]

dbms.connector.<scope>.enabled=true
dbms.connector. <scope>.tls_level=OPTIONAL # Make sure this is set to optional
dbms.ssl.policy. <scope>.enabled=true
dbms.ssl.policy. <scope>.client_auth=NONE # Set to REQUIRE for cluster and backup scopes
dbms.ssl.policy. <scope>.base_directory=/certificates/<scope>
dbms.ssl.policy. <scope>.public_certificate=/certificates/<scope>/public.crt
dbms.ssl.policy. <scope>.private_key=/certificates/<scope>/private.key
dbms.ssl.policy. <scope>.trusted_dir=/certificates/<scope>/trusted
dbms.ssl.policy. <scope>.revoked_dir=/certificates/<scope>/revoked

Neo4j 4.0 Configuration (Directories)

<scope> = [bolt | https | cluster | backup]

$neo4j-home> mkdir –p certificates/<scope>
$neo4j-home> mkdir –p certificates/<scope>/trusted
$neo4j-home> mkdir –p certificates/<scope>/revoked

SSL Providers

The secure networking in Neo4j is provided through the Netty library, which supports both **the native JDK SSL provider(Default)** as well as **Netty-supported OpenSSL derivatives**.

Follow these steps to utilize OpenSSL:

1. Install a suitable dependency into the plugins/ folder of Neo4j. Dependencies can be downloaded from https://netty.io/wiki/forked-tomcat-native.html.

2. Set dbms.netty.ssl.provider=OPENSSL

Using OpenSSL can significantly improve performance, especially for AES-GCM cryptos, e.g. TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256.

Validation & Troubleshooting

Debugging

Validation

- $cypher-shell -u <user> -p <password> -a bolt://my_domain.com:7687 --encryption true
- Enter https://my_domain.com:7473 and click on the lock
- https://www.sslshopper.com/ssl-checker.html. (Enter my_domain.com:7473)
- Make sure you are NOT using IP and only using your DNS name

Neo4j.conf

dbms.jvm.additional=-Djavax.net.debug=ssl:handshake

View & Validate Certificate & key

- $openssl s_client -connect my_domain.com:7687 #bolt
- $openssl s_client -connect my_domain.com:7473 #https
- $nmap --script ssl-enum-ciphers -p <port> <hostname>
- $openssl rsa -in private.key -check
- **PEM** - $openssl x509 -in public.crt –text –noout
- **DER** - $openssl x509 -in certificate.der -inform der -text -noout (<--- only use PEM formatted certificates/keys)

Transformations

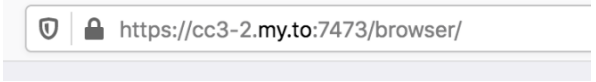
- **PEM to DER:** $openssl x509 -in cert.crt -outform der -out cert.der
- **DER to PEM:** $openssl x509 -in cert.crt -inform der -outform pem -out cert.pem

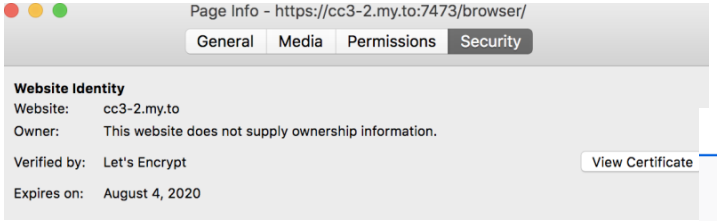
Debug.log

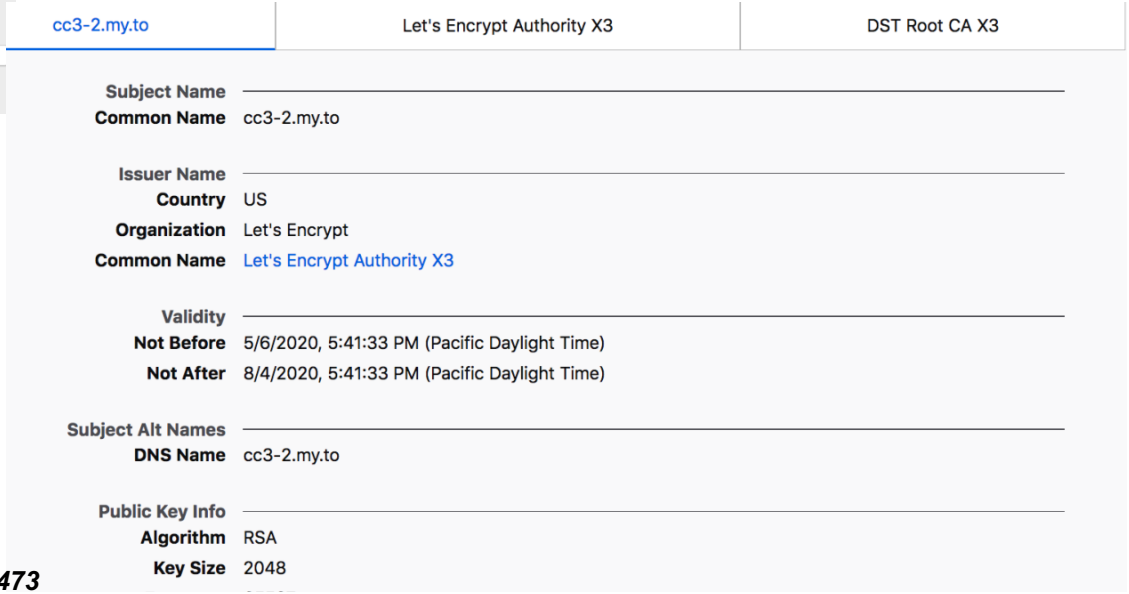
Debug.log

2020-04-05 16:35:18.271+0000 INFO [o.n.k.c.s.SslPolicyLoader]
Loaded SSL policy 'demo' = SslPolicy
{keyCertChain=
Subject: CN=www.neo4j-demo.com, Issuer: CN=Let's Encrypt Authority X3, O=Let's Encrypt, C=US,
Subject: CN=Let's Encrypt Authority X3, O=Let's Encrypt, C=US, Issuer: CN=DST Root CA X3, O=Digital Signature Trust Co.,
ciphers=null,
tlsVersions=[TLSv1.2],
clientAuth=NONE}

View Certificate - Click on the lock







Here you can see the full chain:

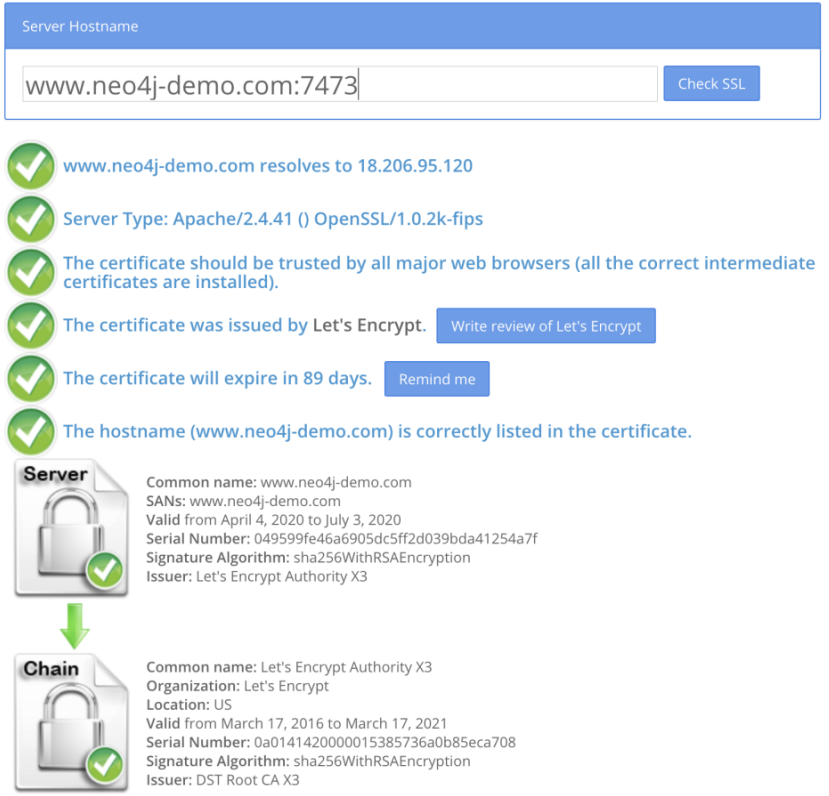
+>DST Root CA X3 (this is the Root certificate)
+----> Let's encrypt authority
+--------> cc3-2.my.to

If the certificate is valid in nginx, apache, or any other application, then the issue should be in your neo4 configuration or usage.

>> Go ahead & try connecting to https://cc3-2.my.to:7473

SSL Checker

- https://www.sslshopper.com/ssl-checker.html



Common Errors:

- Inspect neo4j.log & debug.log for any errors
- When using IP (not domain name) to connect
  - **Web-browser:** SessionExpired: WebSocket connection failure. Due to security constraints in your web browser, the reason for the failure is not available to this Neo4j Driver. Please use your browsers development console to determine the root cause of the failure. Common reasons include the database being unavailable, using the wrong connection URL or temporary network problems. If you have enabled encryption, ensure your browser is configured to trust the certificate Neo4j is configured to use. WebSocket `readyState` is: 3
  - **Cypher-shell**: Failed to establish secured connection with the server
  - 2020-05-05 16:36:37.034+0000 ERROR [o.n.b.t.TransportSelectionHandler] Fatal error occurred when initialising pipeline: [id: 0x000ae28a, L:/127.0.0.1:7687 ! R:/127.0.0.1:36930] javax.net.ssl.SSLHandshakeException: Received fatal alert: **certificate_unknown**
- org.neo4j.server.ServerStartupException: Starting Neo4j failed: Component 'org.neo4j.server.database.LifecycleManagingDatabaseService@7e809b79' was successfully initialized, but failed to start. Please see the attached cause exception **"Base directory '/home/ubuntu/live/certificates/bolt' for SSL policy with name 'bolt' does not exist.".**
- **Backup** encountered invalid certificate on local or remote host:
  - command failed: Failed to run a backup using the available strategies.

View certificates

```
# openssl s_client -connect <hostname>:<port> -showcerts

-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
Server certificate
subject=/CN=localhost
issuer=/CN=localhost
---------
Client Certificate Types: RSA sign, DSA sign, ECDSA sign
...
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-SHA
Server public key is 2048 bit
...
```

Enumerate ciphers

```
# nmap --script ssl-enum-ciphers -p <port> <hostname>

PORT     STATE SERVICE
6000/tcp open  X11
| ssl-enum-ciphers:
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp160k1) - A
|     compressors:
|       NULL
|     cipher preference: indeterminate
|     cipher preference error: Too few ciphers supported
|     warnings:
|       Key exchange parameters of lower strength than certificate key
|_  least strength: A
```

Try connecting with untrusted neo4j

```
Failed to process message on channel [id: 0x73381b0c, L:0.0.0.0/0.0.0.0:7002 !
R:/127.0.0.1:49315]
javax.net.ssl.SSLException: Received fatal alert: certificate_unknown
```

Support Portal KBs

- https://medium.com/neo4j/getting-certificates-for-neo4j-with-letsencrypt-a8d05c415bbd
- https://support.neo4j.com/hc/en-us/articles/360045090054-Resolve-TLS-certificate-errors
- https://support.neo4j.com/hc/en-us/articles/360039722114-TLS-SSL-Configuration-for-Specific-Ciphers
- https://support.neo4j.com/hc/en-us/articles/360040275873-Neo4j-Security-Benchmark
- https://support.neo4j.com/hc/en-us/articles/360023860154-A-demonstration-of-IntraCluster-SSL-Encryption
- https://support.neo4j.com/hc/en-us/articles/360007073994-How-to-set-up-SSL-communcation-when-running-Neo4j-within-a-Docker-Container
- https://aura.support.neo4j.com/hc/en-us/articles/360037562173-Is-My-Connection-to-Neo4j-Aura-Encrypted-
- https://support.neo4j.com/hc/en-us/articles/360019252153-How-Neo4j-Browser-uses-bolt-routing

Thank You!