Kafka - Neo4j - SSL Config

This blogpost provides guidance to configure SSL security between Kafka and Neo4j. This will provide data encryption between Kafka and Neo4j. This does not address ACL confguration inside of KAFKA.

Self Signed Certificates

This section came from https://medium.com/talking-tech-all-around/how-to-enable-and-verify-client-authentication-in-kafka-21e936e670e8
Make sure that you have truststore and keystore JKSs for each server.+ In case you want a self signed certificate, you can use the following commands:

```
mkdir security cd security  export PASSWORD=password keytool -keystore ka
fka.server.keystore.jks -alias localhost -validity 365 -genkey openssl re
q -new -x509 -keyout ca-key -out ca-cert -days 365 keytool -keystore kafk
a.server.truststore.jks -alias CARoot -import -file ca-cert keytool -keys
tore kafka.client1.truststore.jks -alias CARoot -import -file ca-cert key
tool -keystore kafka.server.keystore.jks -alias localhost -certreq -file
cert-file openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -out
cert-signed -days 365 -CAcreateserial -passin pass:$PASSWORD keytool -key
store kafka.server.keystore.jks -alias CARoot -import -file ca-cert keyto
ol -keystore kafka.server.keystore.jks -alias localhost -import -file cer
t-signed keytool -keystore kafka.client1.keystore.jks -alias localhost -v
alidity 365 -genkey keytool -keystore kafka.client1.keystore.jks -alias l
ocalhost -certreq -file cert-file openssl x509 -req -CA ca-cert -CAkey ca
-key -in cert-file -out cert-signed -days 365 -CAcreateserial -passin pas
s:$PASSWORD keytool -keystore kafka.client1.keystore.jks -alias CARoot -i
mport -file ca-cert keytool -keystore kafka.client1.keystore.jks -alias l
ocalhost -import -file cert-signed
```

Once the keystores are created, you have to move the kafka.client1.keystore.jks and kafka.client1.truststore.jks to your neo4j server.

Note: This article discusses addressing this error (Caused by: java.security.cert.CertificateException: No subject alternative names present) that may appear when querying the topic. https://geekflare.com/san-ssl-certificate/

Kafka Configuration

Connect to your Kafka server and modify the config/server.properties file.
This configuration worked for me but I have seen other configurations without the EXTERNAL and INTERNAL settings.
This configuration is for Kafka on AWS but should work for other configurations.

```
listeners=EXTERNAL://0.0.0.0:9092,INTERNAL://0.0.0.0:19092,CLIENT://0.0.0.0:9093,SSL://0.0.0.0:9094 listener.security.protocol.map=EXTERNAL:PLAINTEXT,INTERNAL:PLAINTEXT,CLIENT:PLAINTEXT,SSL:SSL advertised.listeners=EXTERNAL://aws_public_ip:9092,INTERNAL://aws_internal_ip:19092,CLIENT://aws_public_ip:9093,SSL://aws_public_ip:9094 inter.broker.listener.name=INTERNAL  ssl.keystore.location=/home/kafka/security/kafka.server.keystore.jks ssl.keystore.password=neo4jpassword ssl.truststore.location=/home/kafka/security/kafka.server.truststore.jks ssl.truststore.password=neo4jpassword ssl.key.password=neo4jpassword ssl.enabled.protocols=TLSv1.2,TLSv1.1  ssl.endpoint.identification.algorithm=HTTPS ssl.client.auth=required
```

Neo4j Configuration

The following is required for a Neo4j configuration. In this case, we are connecting to the public

AWS IP address. The keystore and truststore locations point to the files that you created earlier in the steps.

Note: The passwords are stored in plaintext so limit access to this neo4j.conf file.

```
kafka.zookeeper.connect=xxx.xxx.xxx.xxx:2181 kafka.bootstrap.servers=xxx.xxx.xxx.xxx:9094 streams.sink.enabled=false streams.sink.polling.interval=1000  streams.source.topic.nodes.neoTest=Person{*}  kafka.auto.offset.reset=earliest kafka.group.id=neo4j streams.sink.dlq=neo4j-dlq  kafka.acks=all kafka.num.partitions=1 kafka.retries=2 kafka.batch.size=16384 kafka.buffer.memory=33554432  kafka.security.protocol=SSL kafka.ssl.truststore.location=/home/ubuntu/security/kafka.client1.truststore.jks kafka.ssl.truststore.password=neo4jpassword kafka.ssl.keystore.location=/home/ubuntu/security/kafka.client1.keystore.jks kafka.ssl.keystore.password=neo4jpassword kafka.ssl.key.password=neo4jpassword kafka.ssl.endpoint.identification.algorithm=HTTPS  dbms.security.procedures.whitelist=apoc.* dbms.security.procedures.unrestricted=apoc.* dbms.jvm.additional=-Djavax.net.debug=ssl:handshake
```

This line **dbms.jvm.additional=-Djavax.net.debug=ssl:handshake** is optional but does help for debugging SSL issues.

Testing

```
After starting Kafka and Neo4j, you can test by creating a Person node in Neo4j and then query the topic as follows:    ./bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic neoTest --from-beginning  If you want to test using SSL, you would do the following:  1. Create a client-ssl.properties file consisting of:
```

```
security.protocol=SSL ssl.truststore.location=/home/kafka/security/kafka.
client.truststore.jks ssl.truststore.password=neo4jpassword ssl.endpoint.
identification.algorithm=
```

2. Query the topic:

./bin/kafka-console-consumer.sh --bootstrap-server 18.217.67.191:9094 --topic neoTest --consume
r.config /home/kafka/client-ssl.properties --from-beginning