Apache Kafka

Kafka Security

Kafka is used in variety of business:

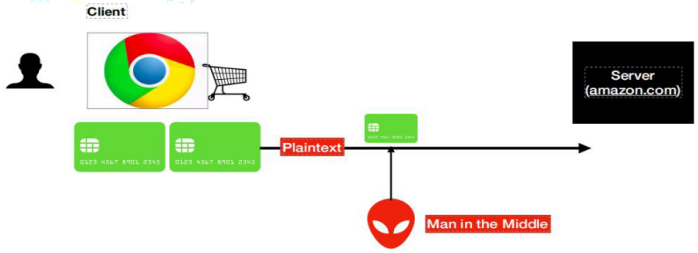
- Banking
- Retail
- Insurance

When the customers are accessing then such web applications and while doing any transactions, the customer might need to enter the sensitive Confidential Information like:

- Credit Card Details
- Customer Information and etc.,

Hence its very important to protect this information from the **man in the middle attacks.** When the transactions were being made using the debit/credit card without the SSL, the senstive information will send to the server as a plain text. As a result, the fraudsters can tamper/steal the sensitive data and they can perform the transactions on behalf of the actual user.

Without SSL Encryption:



To avoid such man in the middle attacks, Kafka is offering the security using the following two popular protocols:

- ✓ SSL (Secured Socket Layer), which is TLS (Transport Layer Security) now
- ✓ SASL (Simple Authentication and Security Layer)

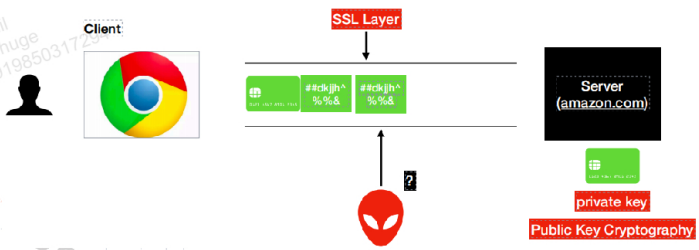
Secured Socket Layer(SSL)

Apache Kafka

SSL used for two things:

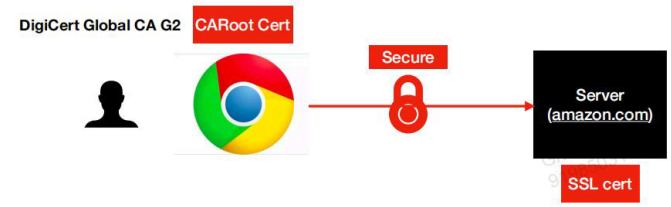
- ✓ Encryption
- ✓ Authentication

With SSL Encryption:



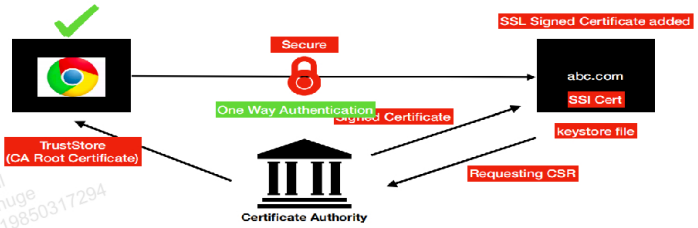
SSL Authentication is to make sure we are talking to the correct server

SSL Authentication Working:



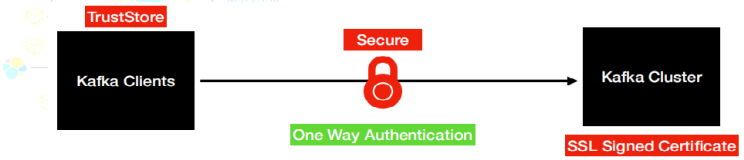
SSL Requests by Enterprise:

Apache Kafka



Kafka SSL Setup:

Securing the Kafka Cluster with the SSL includes the following components. We have to create the SSL Self Signed Certificate at the Kafka cluster side and a trust store need to be share with the Kafka clients.



SSL Set Up Steps:

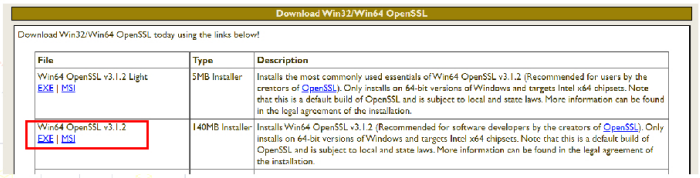
1. Generate **server.keystore.jks**
2. SetUp Local Certificate Authority
3. Create CSR(Certificate Signing Request)
4. Sign the SSL Certificate
5. Add the Signed SSL Certificate to **server.keystore** file
6. Configure the **SSL cert** in our Kafka Broker
7. Create **client.truststore.jks** for the client

Apache Kafka

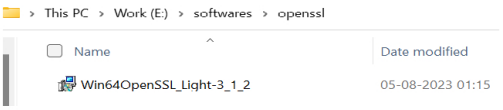
Local Certificate Authority

Download and setup the openssl:

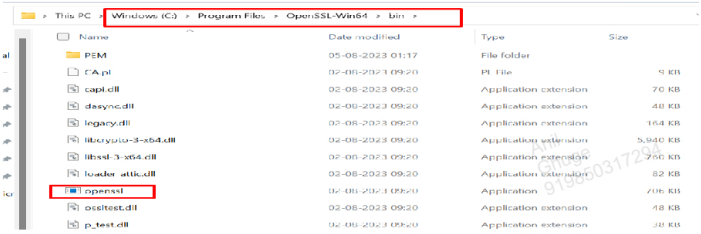
Step-1: https://slproweb.com/products/Win32OpenSSL.html



Step-2:

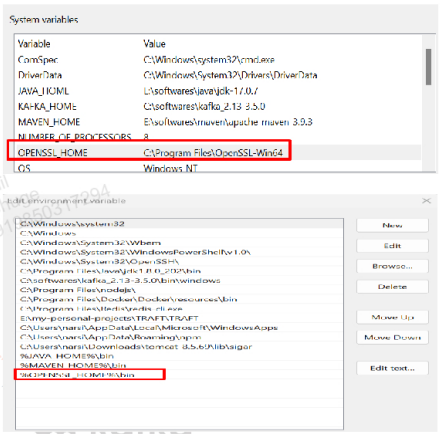


Step-3:



Step-4:

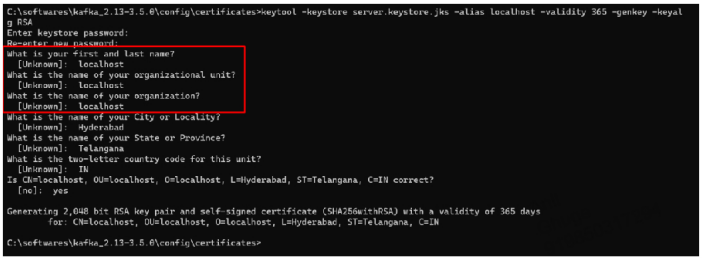
Apache Kafka



Step-1: Generate server.keystore.jks

The below command is to generate the keyStore. KeyStore in general has information about the server and the organization.

```
keytool -keystore server.keystore.jks -alias localhost -validity 365 -genkey -keyalg RSA
```



Note: what is your first and last name? This is always the domain name. In our case it is **localhost**.

Example: After entering all the details the final value will look like below

Example: After entering all the details the final value will look like below:

Apache Kafka

```
CN=localhost, OU=localhost, O=localhost, L=Hyderabad, ST=TN, C=IN
```

Step-2: Setup Local Certificate Authority

The below command will generate the ca cert(SSL cert) and private key. This is normally needed if we are self signing the request.

```
openssl req -new -x509 -keyout ca-key -out ca-cert -days 365 -subj "/CN=local-security-CA"
```

Step-3: Create CSR(Certificate Signing request)

The below command will create a cert-file as a result of executing the command.

```
keytool -keystore server.keystore.jks -alias localhost -certreq -file cert-file
```

Step-4: Sign the SSL Certificate

The below command takes care of signing the CSR and then it spits out a file cert-signed.

```
openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -out cert-signed -days 365 -CAcreateserial -passin pass:password
```

To view the content inside the file cert-signed, run the below command.

```
keytool -printcert -v -file cert-signed
```

Step-5:Add the Signed SSL Certificate to server.keystore file

```
keytool -keystore server.keystore.jks -alias CARoot -import -file ca-cert

keytool -keystore server.keystore.jks -alias localhost -import -file cert-signed
```

Step-6: Configure the SSL cert in our Kafka Broker

In server.properties file at the kafka cluster brokers we need to configure the following peoprties.

Apache Kafka

```
#    listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://:9092,SSL://localhost:9095
```

```
ssl.keystore.location=<location>/server.keystore.jks

ssl.keystore.password=password

ssl.key.password=password

ssl.endpoint.identification.algorithm=
```

Step-7: Create client.truststore.jks for the client

The below command takes care of generating the truststore for us and adds the CA-Cert in to it. This is to make sure the client is going to trust all the certs issued by CA.

```
keytool -keystore client.truststore.jks -alias CARoot -import -file ca-cert
```

Accessing SSL Enabled Topics using Console Producers/Consumers

Step-1: Create a topic

```
./kafka-topics.sh --create --topic test-topic -zookeeper localhost:2181 --replication-factor 1 --partitions 3
```

Step-2: Create a file named client-ssl.properties and have the below properties configured in there.

```
security.protocol=SSL

ssl.truststore.location=<location>/client.truststore.jks

ssl.truststore.password=password

ssl.truststore.type=JKS
```

Step-3: Producing Messages to Secured Topic. The below Command to Produce Messages to the secured topic

```
kafka-console-producer.bat --broker-list localhost:9095,localhost:9096,localhost:9097 --topic test-
```

Apache Kafka

```
topic --producer.config client-ssl.properties
```

Step-4: Consuming Messages from a Secured Topic. Command to Produce Messages to the secured topic

```
kafka-console-consumer.bat --bootstrap-server localhost:9095,localhost:9096,localhost:9097 --topic test-topic --consumer.config client-ssl.properties
```

Step-5: Producing Messages to Non-Secured Topic

```
./kafka-console-producer.sh --broker-list localhost:9092,localhost:9093,localhost:9094 --topic test-topic
```

Step-6: Consuming Messages from a Non-Secured Topic

```
./kafka-console-consumer.sh --bootstrap-server localhost:9092,localhost:9093,localhost:9094 --topic test-topic
```

2 Way SSL Authentication

Step-1: This config is to enable the client authentication at the cluster end.

```
keytool -keystore server.truststore.jks -alias CARoot -import -file ca-cert
```

Step-2: Add the ssl.client.auth property in the server.properties.

```
ssl.truststore.location=<location>/server.truststore.jks
```

Apache Kafka

```
ssl.truststore.password=password

ssl.client.auth=required
```

Step-3: Kafka Client should have the following the config in the client-ssl.properties file

```
ssl.keystore.type=JKS

ssl.keystore.location=<location>/client.keystore.jks

ssl.keystore.password=password

ssl.key.password=password
```