Published in Egen Engineering & Beyond

Vishwa Teja Vangari Follow

Jun 23, 2019 · 7 min read · Listen

Securing Apache Kafka Cluster using SSL, SASL and ACL

Pre-requisite: Novice skills on Apache Kafka, Kafka producers and consumers.
This blog will focus more on SASL, SSL and ACL on top of Apache Kafka Cluster.

Securing Apache Kafka Cluster

In this blog, we will go over the configurations for enabling authentication using **SCRAM** , authorization using ***SimpleAclAuthorizer*** and encryption between clients and server using **SSL.**

Apache Kafka these days is widely used for exchange of sensitive information as well between various systems with in the company and also between different companies. So secure connections like HTTPS is essential between client applications and Kafka Server. Apache Kafka open source community contributed multiple Kafka Security options for Authentication, Authorization and Encryption.

Authentication in Kafka:

- SSL
- SASL: PLAIN, SCRAM(SHA-256 and SHA-512), OAUTHBEARER, GSSAPI(Kerberos)

Authorization in Kafka: Kafka comes with simple authorization class *kafka.security.auth.SimpleAclAuthorizer* for handling ACL's (create, read, write,


Encryption: Data Encryption over the network between clients and Kafka server. i.e between producers and Kafka server, consumers and Kafka server.

Authentication using SCRAM:

Salted Challenge Response Authentication Mechanism (SCRAM) is similar to authentication using username/password. Apache Kafka supports `SCRAM-SHA-256` and `SCRAM-SHA-512`. For this mechanism, Kafka by default (*ScramLoginModule*) stores SCRAM credentials in zookeeper with the salt, so zookeeper need to be secured in the private network with very limited access. The default iteration count of 4096 is used if iterations are not specified. A random salt is created and the SCRAM identity consisting of salt, iterations, StoredKey and ServerKey are stored in ZooKeeper. In our example below, we will use `SCRAM-SHA-512` for authentication.

Authorization using SimpleAclAuthorizer:

Apache Kafka ships out with simple default authorization implementation ***kafka.security.auth.SimpleAclAuthorizer*** for all ACL operations. *SimpleAcl Authroizer* comes up create, read, write, cluster_action, alter_configs, describe

Encryption using SSL:

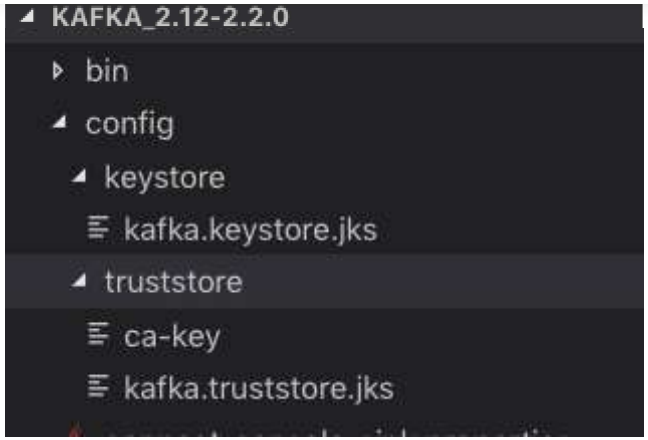
SSL(Secure Sockets Layer) can be configured for encryption and also serves as 2-way authentication between client and server. i.e broker authenticates client using client truststore certificate and client authenticates broker using broker truststore certificate. In this article, we will just configure SSL only for encryption.SSL uses private-key/certificates pairs which are used during the SSL handshake process.

- *each Kafka broker needs its own private-key/certificate pair, and then client uses certificate to authenticate the broker*

You could refer this Github repo for spinning up secure cluster by adding the required configs locally on your machine.
Now, Let's get onto working example:

Download Apache Kafka binary from open source Apache Kafka Downloads.

SSL Certificate and Key generati broker SSL keystore and truststore certificate using confluent-platform-security-tools generate ssl



keystore and truststore in config folder

After generating truststore and keystore folders through this script, we will copy them over to config folder in downloaded Kafka binary directory.

We will execute all below commands by navigating to <kafka-binary-dir>/ in terminal.

Start Zookeeper

```
./bin/zookeeper-server-start.sh config/zookeeper.properties
```

```
./bin/kafka-configs.sh --zookeeper localhost:2181 --alter --add-config
'SCRAM-SHA-512=[password='admin-secret']' --entity-type users --
entity-name admin

Completed Updating config for entity: user-principal 'admin'.
```

Create kafka_server_jaas.conf file in config folder

```
KafkaServer {
org.apache.kafka.common.security.scram.ScramLoginModule required
username="admin"
password="admin-secret";
};
```

Create ssl-user-config.properties file in config folder

```
security.protocol=SASL_SSL
sasl.mechanism=SCRAM-SHA-512
sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModu
le required username="demo-user" password="secret";
ssl.truststore.location=
```


Append below security properties to existing Kafka server.properties file in config folder.

```
########### SECURITY using SCRAM-SHA-512 and SSL ###################

listeners=PLAINTEXT://localhost:9092,SASL_PLAINTEXT://localhost:9093,S
ASL_SSL://localhost:9094

advertised.listeners=PLAINTEXT://localhost:9092,SASL_PLAINTEXT://local
host:9093,SASL_SSL://localhost:9094

security.inter.broker.protocol=SASL_SSL

ssl.endpoint.identification.algorithm=

ssl.client.auth=required

sasl.mechanism.inter.broker.protocol=SCRAM-SHA-512

sasl.enabled.mechanisms=SCRAM-SHA-512

# Broker security settings
ssl.truststore.location=
<kafka-binary-dir>/config/truststore/kafka.truststore.jks

ssl.truststore.password=password

ssl.keystore.location=
<kafka-binary-dir>/config/keystore/kafka.keystore.jks

ssl.keystore.password=password
```


```
#zookeeper SASL
zookeeper.set.acl=false
########### SECURITY using SCRAM-SHA-512 and SSL ###################
```

Now Start Kafka with jaas conf file:

```
export KAFKA_OPTS=-Djava.security.auth.login.config=<kafka-binary-
dir>/config/kafka_server_jaas.conf

./bin/kafka-server-start.sh ./config/server.properties
```

Clients can now connect to Kafka cluster using 3 ways for above server.properties configuration.

- ***localhost:9092:*** PLAIN_TEXT with no authentication and authorization.
- ***localhost:9093:*** PLAIN_TEXT with authentication and authorization.
- ***l*ocalhost:9094:** along with Encryption, authentication, authorization.

Now Let’s go over different workflow’s using Kafka Cluster.


```
./bin/kafka-configs.sh --zookeeper localhost:2181 --alter --add-config
'SCRAM-SHA-512=[password='secret']' --entity-type users --entity-name
demouser

Completed Updating config for entity: user-principal 'demouser'.
```

*Create **ssl-user-config.properties** in /config folder:*

```
security.protocol=SASL_SSL

sasl.mechanism=SCRAM-SHA-512

sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModu
le required username="demouser" password="secret";

ssl.truststore.location=
<kafka-bin-dir>/config/truststore/kafka.truststore.jks

ssl.truststore.password=password
```

Creating Topic without Create Permissions:


```
authorized to access topics: [Topic authorization failed.]
[2019-06-20 11:27:29,244] ERROR
java.util.concurrent.ExecutionException:
org.apache.kafka.common.errors.TopicAuthorizationException: Not
authorized to access topics: [Topic authorization failed.]
 at
org.apache.kafka.common.internals.KafkaFutureImpl.wrapAndThrow(KafkaFu
tureImpl.java:45)
 at
org.apache.kafka.common.internals.KafkaFutureImpl.access$000(KafkaFutu
reImpl.java:32)
 at
org.apache.kafka.common.internals.KafkaFutureImpl$SingleWaiter.await(K
afkaFutureImpl.java:89)
 at
org.apache.kafka.common.internals.KafkaFutureImpl.get(KafkaFutureImpl.
java:260)
 at
kafka.admin.TopicCommand$AdminClientTopicService.createTopic(TopicComm
and.scala:175)
 at
kafka.admin.TopicCommand$TopicService.createTopic(TopicCommand.scala:1
34)
 at
kafka.admin.TopicCommand$TopicService.createTopic$(TopicCommand.scala:
129)
 at
kafka.admin.TopicCommand$AdminClientTopicService.createTopic(TopicComm
and.scala:157)
 at kafka.admin.TopicCommand$.main(TopicCommand.scala:60)
 at kafka.admin.TopicCommand.main(TopicCommand.scala)
Caused by: org.apache.kafka.common.errors.TopicAuthorizationException:
```


*Assign **Create and Describe** permissions to **demouser** and then **create** topic:*

```
./bin/kafka-acls.sh --authorizer-properties
zookeeper.connect=localhost:2181 --add --allow-principal User:demouser
--operation Create --operation Describe  --topic demo-topic

Adding ACLs for resource `Topic:LITERAL:demo-topic`:
  User:demouser has Allow permission for operations: Create from
hosts: *
 User:demouser has Allow permission for operations: Describe from
hosts: *

Current ACLs for resource `Topic:LITERAL:demo-topic`:
  User:demouser has Allow permission for operations: Create from
hosts: *
 User:demouser has Allow permission for operations: Describe from
hosts: *

./bin/kafka-topics.sh --create --bootstrap-server localhost:9094 --
command-config ./config/ssl-user-config.properties --replication-
factor 1 --partitions 1 --topic demo-topic

Topic demo-topic created
```

Create ssl-producer.properties in config folder.

```
security.protocol=SASL_SSL

sasl.mechanism=SCRAM-SHA-512

sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModu
le required username="demouser" password="secret";

ssl.truststore.location=
<kafka-binary-dir>/config/truststore/kafka.truststore.jks

ssl.truststore.password=password
```

Produce with incorrect password: *By modifying password to something else in ssl-producer.properties and then try to produce data on* ***demo-topic***

```
./bin/kafka-console-producer.sh --broker-list localhost:9094 --topic
demo-topic --producer.config config/ssl-producer.properties

[2019-06-22 18:41:27,406] ERROR [Producer clientId=console-producer]
Connection to node -1 (localhost/127.0.0.1:9094) failed authentication
due to: Authentication failed during authentication due to invalid
credentials with SASL mechanism SCRAM-SHA-512
(org.apache.kafka.clients.NetworkClient)
```

Produce with correct password *and with no produce permissions on topic* ***demo-***


```
./bin/kafka-console-producer.sh --broker-list localhost:9094 --topic
demo-topic --producer.config config/ssl-producer.properties

>ERROR Error when sending message to topic demo-topic with key: null,
value: 8 bytes with error:
(org.apache.kafka.clients.producer.internals.ErrorLoggingCallback)
org.apache.kafka.common.errors.TopicAuthorizationException: Not
authorized to access topics: [demo-topic]
```

Assign Producer permissions for demouser to demo-topic

```
./bin/kafka-acls.sh --authorizer-properties
zookeeper.connect=localhost:2181 --add --allow-principal User:demouser
--producer --topic demo-topic

Adding ACLs for resource `Topic:LITERAL:demo-topic`:
  User:demouser has Allow permission for operations: Create from
hosts: *
 User:demouser has Allow permission for operations: Describe from
hosts: *
 User:demouser has Allow permission for operations: Write from hosts:
*

Current ACLs for resource `Topic:LITERAL:demo-topic`:
  User:demouser has Allow permission for operations: Create from
hosts: *
 User:demouser has Allow permission for operations: Describe from
```


*Produce messages on **demo-topic** with correct password and permissions:*

```
./bin/kafka-console-producer.sh --broker-list localhost:9094 --topic
demo-topic --producer.config config/ssl-producer.properties
>message1
>message2
>message3
^C
```

we have now produced 3 messages on to demo-topic, let's try to consume those messages from this topic, by creating a consumer.

Create ssl-consumer.properties in config folder

```
bootstrap.servers=localhost:9094

# consumer group id

group.id=demo-consumer-group

### SECURITY ######
security.protocol=SASL_SSL
```


```
ssl.truststore.location=
<kafka-binary-dir>/config/truststore/kafka.truststore.jks

ssl.truststore.password=password
```

Consume data from ***demo-topic*** *with incorrect password in ssl-consumer.properties*

```
./bin/kafka-console-consumer.sh --bootstrap-server localhost:9094 --
topic demo-topic --from-beginning --consumer.config config/ssl-
consumer.properties

[2019-06-22 18:46:41,983] ERROR Error processing message, terminating
consumer process:  (kafka.tools.ConsoleConsumer$)
org.apache.kafka.common.errors.SaslAuthenticationException:
Authentication failed during authentication due to invalid credentials
with SASL mechanism SCRAM-SHA-512
Processed a total of 0 messages
```

Consume data from ***demo-topic*** *with* ***correct password*** *in ssl-consumer.properties*

```
org.apache.kafka.common.errors.GroupAuthorizationException: Not
authorized to access group: demo-consumer-group
Processed a total of 0 messages
```

Assign Consumer permissions for *demouser* on *demo-topic*

```
./bin/kafka-acls.sh --authorizer-properties
zookeeper.connect=localhost:2181 --add --allow-principal User:demouser
--consumer --topic demo-topic --group demo-consumer-group

Adding ACLs for resource `Topic:LITERAL:demo-topic`:
  User:demouser has Allow permission for operations: Describe from
hosts: *
 User:demouser has Allow permission for operations: Read from hosts: *

Adding ACLs for resource `Group:LITERAL:demo-consumer-group`:
  User:demouser has Allow permission for operations: Read from hosts:
*

Current ACLs for resource `Topic:LITERAL:demo-topic`:
  User:demouser has Allow permission for operations: Create from
hosts: *
 User:demouser has Allow permission for operations: Describe from
hosts: *
 User:demouser has Allow permission for operations: Write from hosts:
*
 User:demouser has Allow permission for operations: Read from hosts: *
```


Consume messages** from **demo-topic** with **correct password** and **consume permissions:

```
./bin/kafka-console-consumer.sh --bootstrap-server localhost:9094 --
topic demo-topic --from-beginning --consumer.config config/ssl-
consumer.properties
message1
message2
message3
^C
```

Summary

To summarize, we have seen how to encrypt Kafka Cluster connections using SSL, authentication using **SCRAM**, and authorization using a Kafka inbuilt ***SimpleAclAuthorizer*** class. In the upcoming blogs, I will show you how to configure other options such as authentication using the OAuthBearer Token.

If you find this blog helpful, be sure to give it a few claps, read more or follow me on LinkedIn


- https://kafka.apache.org/documentation/#security
- https://docs.confluent.io/current/security/index.html