KIP-163: Lower the Minimum Required ACL Permission of OffsetFetch

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: **KAFKA-4585** - Getting issue details... STATUS

Released: 1.0.0

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Note: The discussion in this KIP applies to Java based (new) consumer only as the security feature is not supported by the old consumer.

From an authorization and ACL point of view, three operations (permission types) are defined for consumer groups: *Describe*, *Read*, *All*. By default, *Read* implies *Describe*, and *All* implies all the other operations.

Current consumer group related APIs and their minimum required permissions are listed in the following table:

API	Minimum Required Permission
`DescribeGroup`	*Describe* (Group)
`FindCoordinator`	*Describe* (Group)
`Heartbeat`	*Read* (Group)
`JoinGroup`	*Read* (Group)
`LeaveGroup`	*Read* (Group)
`ListGroup`	*Describe* (Cluster)
`OffsetCommit`	*Read* (Group)
`OffsetFetch`	*Read* (Group)
`SyncGroup`	*Read* (Group)
`AddOffsetsToTxn`	*Read* (Group)
`TxnOffsetCommit`	*Read* (Group)

The pattern we can see in this table is that a minimum *Read* permission is used for mutating APIs, whereas a minimum *Describe* permission is used for non-mutating APIs. One exception to this pattern is `OffsetFetch`, which is a non-mutating API, but requires a *Read* access. A *Read* access requirement for `OffsetFetch` is too restrictive, and unnecessary. Consider the following example by @ewencp in the corresponding JIRA's description: If we want to write a tool that only monitors offsets (no commits), we cannot achieve it with the current ACL settings. Because accessing the `OffsetFetch` API requires a *Read* permission; but a *Read* permission means we are also authorized to use the `CommitOffset` API (side note: for this tool to be able to read offsets of a group, it needs to have *Describe* access to the topics the group is consuming from. In other words, the tool will be able to see offsets of all topics (topic partitions) in the group it has *Describe* access to).

The other, and perhaps more compelling, incentive for this change is that the current ACL settings breaks a certain functionality (and this functionality seems to have been broken for a while). As mentioned in the above table the minimum required permission for `DescribeGroup` and `OffsetFetch` is *Describe* and *Read*, respectively. But implementation of the describe group command line makes use of `OffsetFetch` API (version 0 and 1 pre-KIP-88, and version 2 post-KIP-88). Therefore, a user who is granted the current minimum requirement permission *Describe* for `DescribeGroup` still would not be able to run the describe group command and get the expected result. They would see something like this in the output:

```
Error: Executing consumer group command failed due to Not authorized to access group: Group authorization
failed.
```

If we make the change suggested in the next section, the command runs successfully and reports the group offsets.

The following potential unit tests in `scala.integration.kafka.api.AuthorizerIntegrationTest` could further clarify the problem.

```
// this test is to clarify that the issue exists for the consumer group command line only, and not the API
@Test
def testDescribeGroupApiWithGroupDescribe() {
  addAndVerifyAcls(Set(new Acl(KafkaPrincipal.ANONYMOUS, Allow, Acl.WildCardHost, Describe)), groupResource)
  addAndVerifyAcls(Set(new Acl(KafkaPrincipal.ANONYMOUS, Allow, Acl.WildCardHost, Describe)), topicResource)
  AdminClient.createSimplePlaintext(brokerList).describeConsumerGroup(group)
}

// this test highlights the issue with command line, where the supposedly sufficient 'Describe' access is not
enough to run the command
@Test(expected = classOf[GroupAuthorizationException])
def testDescribeGroupCliWithGroupDescribe() {
  addAndVerifyAcls(Set(new Acl(KafkaPrincipal.ANONYMOUS, Allow, Acl.WildCardHost, Describe)), groupResource)
  addAndVerifyAcls(Set(new Acl(KafkaPrincipal.ANONYMOUS, Allow, Acl.WildCardHost, Describe)), topicResource)

  val cgcArgs = Array("--bootstrap-server", brokerList, "--describe", "--group", group)
  val opts = new ConsumerGroupCommandOptions(cgcArgs)
  val consumerGroupService = new KafkaConsumerGroupService(opts)
  consumerGroupService.describeGroup()
}

// this test confirms that a minimum of 'Read' access is required to successfully run the command
@Test
def testDescribeGroupCliWithGroupRead() {
  addAndVerifyAcls(Set(new Acl(KafkaPrincipal.ANONYMOUS, Allow, Acl.WildCardHost, Read)), groupResource)
  addAndVerifyAcls(Set(new Acl(KafkaPrincipal.ANONYMOUS, Allow, Acl.WildCardHost, Describe)), topicResource)

  val cgcArgs = Array("--bootstrap-server", brokerList, "--describe", "--group", group)
  val opts = new ConsumerGroupCommandOptions(cgcArgs)
  val consumerGroupService = new KafkaConsumerGroupService(opts)
  consumerGroupService.describeGroup()
}
```

Proposed Changes

The change proposed by this KIP is very simple: to lower the minimum required permission of the `OffsetFetch` API from *Read* to *Describe*. These minimum required permissions are hard-coded in `kafka.server.KafkaApis.scala` inside each API handler method. For example, the part that enforces the minimum required permission for the `OffsetFetch` API currently looks like this:

```
if (!authorize(request.session, Read, new Resource(Group, offsetFetchRequest.groupId)))
          offsetFetchRequest.getErrorResponse(requestThrottleMs, Errors.GROUP_AUTHORIZATION_FAILED)
```

And the proposal is to to modify it to:

```
if (!authorize(request.session, Describe, new Resource(Group, offsetFetchRequest.groupId)))
          offsetFetchRequest.getErrorResponse(requestThrottleMs, Errors.GROUP_AUTHORIZATION_FAILED)
```

Compatibility, Deprecation, and Migration Plan

- A user that already has *Read* permission to a consumer group, with this change, would still be able to query the group like before (*Read* implies *D escribe*). For this user the change is backward compatible.

- Consider a user with *Describe* access. The group *Describe* access implies access to `DescribeGroup` and `FindCoordinator` APIs; even though this user cannot make use of `DescribeGroup`, as explained above. Giving this user access to `OffsetFetch` API means fixing that broken experience.

In general, As a result of this change, Kafka admins may need to revisit the relevant ACLs and update them if necessary.

Rejected Alternatives