BP-3 - Security support

Problem

We want to secure BookKeeper clusters, so at least we have to cover this main issues:

- Use ACLs while writing to ZooKeeper
- Provide an extensible Authentication framework (in BookKeeper 4.4.0 there is already partial support for this)
  - Provide a out-of-the box plugin which implements standard SASL authentication, with at least support for GSSAPI/Kerberos and DIGEST-MD5 mechs
- Provide initial support for Authorization
- SSL/TLS support for encrypting comunications and/or to implement client authentication

Proposals

Major concers are about protocol compatibility, data encryption, configuration on Bookie side and client-side, rolling upgrades and ZooKeeper security.

Secure ZooKeeper data structures

In order to implement a secured ZooKeeper layout at least we have to set ACLs which prevent unknown users to modify and/or access ledgers metadata and bookies metadata.

The idea is to have a global "secure zookeeper nodes" (zkEnableSecurity), to be set both on bookies and on clients, as bookkeeper clients do the most of the work actually.

The initial proposal is to set the default ZooKeeper ACL ZooDefs.Ids.CREATOR_ALL_ACL to every Z-Node created from BookKeeper clients and Bookies.

This will work fine only if every client and bookie performs authentication to ZooKeeper with the same "user".

This is a very common practice in the Hadoop ecosystem (see Kafka, HBase...) and with Kerberos based setups.

Usually in a Kerberized ZooKeeper setup each machine logs in using a principal in the form zkclient/HOSTNAME@REALM and ZooKeeper strips out the HOSTNAME and REALM part: this way each client gets authenticated as simply "zkclient" user.

Changes on protocol for AuthMessage message:

On 4.4.0 there is a issue on the BookieAuthProvider and ClientAuthProvider classes, these classes depend directly on the AuthMessage class which in turn depends on protobuf, which is a shaded dependency.

The idea is to remove the usage of that class on the public API and make the implementor use a standard AuthToken class which is a simple wrapper for an array of bytes.

Common Authentication frameworks, like SASL, need only to exchange a sequence of arrays of bytes, and so this new API would be suitable for most of the usecases.

On 4.4.0 the AuthMessage message is defined to be extensible, such a feature will not be needed any more.

ISSUE: can we drop the 'extensions' line on the definition of AuthMessage in BookkeeperProtocol.proto ?

this message

```
message AuthMessage {
    required string authPluginName = 1;
    extensions 1000 to max;
}
```

will become

```
message AuthMessage {
    required string authPluginName = 1;
    required bytes  token = 2;
}
```

Rolling upgrade to switch from authentication types:

The proposal is to support the switch from "no-auth" to any plugin is the following:

1) when no auth is configured on bookie-side the default "AuthDisabledPlugin" plugin comes to play (on 4.4.0 we call it "NULLPlugin")

2) when no auth is configured on bookies the bookie goes to a permissive mode in which any client is allowed to connect and use bookie functions, any 'auth' message will be answered with an OK status code from the AuthDisabledPlugin

3) when no auth is configured on bookies each attempt of authentication by the client will receive the OK status code from the AuthDisabledPlugin, and so the client will understand that the bookie is not performing authentication, and goes to the authenticated state

A rolling upgrade from AuthDisabledPlugin to any other plugin can be implemented by rebooting clients with the new plugin: this way bookies will authenticate clients with AuthDisabledPlugin and clients will go to authenticated state anyway.

Then you will restart bookies with the new plugin and clients will perform real authentication.

Bookie-to-Bookie comunications will follow the same flow during the rolling upgrade, because each bookie can be a client for the other bookies and will authenticate thru the AuthDisabledPlugin flow or to the requested plugin flow.

Attaching the authenticated user to the server-side peer:

Another issue is to attach to the server-side view of the connection (there is no specific class, it is the BookiePipeline instance created for each client) the user-id (principal) of the client which performed authentication, this will be the base for ACLs, quotas, auditing and further security-related features.

A change to the AuthProvider interface will be needed, at least the AuthHandshakeCompleteCallback needs to receive the user-id upon successful authentication.

from:

```
class AuthHandshakeCompleteCallback implements GenericCallback<Void> {
  @Override
  public void operationComplete(int rc, Void v) {
    if (rc == BKException.Code.OK) {
       authenticated = true;
    } else {
        LOG.debug("Authentication failed on server side");
    }
    }
}
```

to something like:

```
class AuthHandshakeCompleteCallback implements
GenericCallback<BookKeeperPrincipal> {
  @Override
  public void operationComplete(int rc, BookKeeperPrincipal userId) {
    if (rc == BKException.Code.OK) {
       authenticated = true;
       authorizedId = userId;
    } else {
        LOG.debug("Authentication failed on server side");
    }
    }
}
```

The "authorizedId" field of AuthHandler.ServerSideHandler need to be pushed back to the BookiePipeline.

Thinking about SSL mutual autehntication it wold be useful to change the BookieAuthProvider.Factory#newProvider method.

In 4.4.0 this method receives only the InetSocketAddress of the client connection, it would be better to pass a structure contains:

- the InetSocketAddress of the client (maybe it would be a SocketAddress, in order to deal better with "Local" connections)
- Certificates and other kind of Principals attached to the connection

It is better not to hard-code the dependency on Netty on such an interface, it would be enough to create a simple bean for 4.5.0 like:

```
class PeerInfo {
     SocketAddress address;
     List<Object> principals;
}
```

It would be better to add this feature on 4.5.0 as we are breaking compatibility with 4.4 Authentication framework

The final signature would be:

```
 BookieAuthProvider newProvider(PeerInfo peerInfo,

GenericCallback<BookKeeperPrincipal> completeCb);
```

ISSUE: do we need a BookKeeperPrincipal class or it is enough to use a simple java.lang.String ?

Some projects, like Kafka use a common class (KakfaPrincipal, see KIP-11 - Authorization Interface) in order to provide better support for the other features

SASL/Kerberos support:

The initial proposal is to support SASL authentiaction as ZooKeeper does, this is turn is implemented using standard Java support for JAAS and SASL

We can implement an AuthPlugin which uses SASL (see https://github.com/eolivelli/bookkeeper/tree/BOOKKEEPER-391/bookkeeper-server/src/main/java/org/apache/bookkeeper/sasl as a proof-of-concept implementation) and bundle it on standard BookKeeper distribution.

The admin needs to simply write a JAAS configuration file like this for Kerberos and setup keytabs for each machine.

```
Bookie {
       com.sun.security.auth.module.Krb5LoginModule required debug=true
       useKeyTab=true
       keyTab=/path/to/server.keytab
       storeKey=true
       useTicketCache=false
       principal=bookkeeper/HOSTNAME@REALM
};

BookKeeper {
       com.sun.security.auth.module.Krb5LoginModule required debug=true
       useKeyTab=true
       keyTab=/path/to/client.keytab
       storeKey=true
       useTicketCache=false
       principal=username/HOSTNAME@REALM
};

Auditor {
       com.sun.security.auth.module.Krb5LoginModule required debug=true
       useKeyTab=true
       keyTab=/path/to/server.keytab
       storeKey=true
       useTicketCache=false
       principal=bookkeeper/HOSTNAME@REALM
};
```

ZookKeeper enable the usage of JAAS to configure SASL based DIGEST-MD5 authentication, this is very useful for little deployments and for testing purposes. See ZooKeeper documentation for better explanations. Kakfa supports a similar configuration for simple JAAS/SASL setups.

For simple DIGEST-MD5 login ZooKeeper uses JAAS files like these:

```
Bookie {
       org.apache.zookeeper.server.auth.DigestLoginModule required
       user_user1="testpwd";
       user_user2="testpwd";
       user_auditor="testpwd";
 };

BookKeeper {
       org.apache.zookeeper.server.auth.DigestLoginModule required
       username="user1"
       password="testpwd";
};

Auditor {
       org.apache.zookeeper.server.auth.DigestLoginModule required
       username="auditor"
       password="testpwd";
};
```

Following the conventions the "Bookie" principal will be bookeeper/HOSTNAME@REALM (for instance in zookeeper it has to be zookeeper /HOSTNAME@REALM and for kafka it is kafka/HOSTNAME@REALM).

The authorized id (user) will be the Principal attached to the JAAS subject, so bookeeper/HOSTNAME@REALM. In ZooKeeper there are options to strip out the HOSTNAME part and the REALM part (see kerberos.removeHostFromPrincipal, kerberos.removeRealmFromPrincipal properties).

ISSUE: In order to support a distinct JAAS configuration entry for the Auditor (Bookie-to-Bookie comunications) the special BookKeeper instance used by the Auditor must be configured in order not to use the standard "BookieClient" entry. Maybe we can use an internal configuration option handled by BookKeeper client like 'auditorMode'. In this mode the entry will be choosen following the Auditor conventions.

ISSUE: On ZookKeeper the SASL mechanism is decided upon the type of JAAS Subject, this is very simply from admin to be configured. We should make the configuration more explicit, something like a configuration property sasl.mech=GSSAPI|DIGEST-M5.....

SSL/TLS Support

The initial proposal is enable TLS support using the StartTLS mechanism, this way Bookie will continue to advertise only one network endpoint, which in turn is effectively used as Bookie ID.

Having a Bookie with more than one endpoint will need more metadata refactoring, the StartTLS proposal let us skip this change for this version.

In order to support TLS we have to implement TLS comunications on Netty 3 (maybe on Netty 4) and add a new protocol message to implement the StartTLS.

We can implement an AuthPlugin which will use the Certificate sent from the client and attach it to the server-side connection peer.

We can add an optional scheduled task which checks certificate validity, this can be done inside the AuthPlugin, but the AuthPlugin will need to hold a reference to an "handle" to the underlying connection, in order to shutdown it in case of certificate expiration

In order to support such a Connection Handle the PeerInfo structure passed to the AuthPlugin should be changed and become an active object

```
class ServerSideConnectionHandle {
     SocketAddress remoteAddress;
     List<Object> principals;
     void closeConnection();
}
```

The SSL Authentication plugin will retain references to every authenticated connection and we need to be handle carefully such references in order not to get into resource leaks

Action

- ~~BOOKKEEPER-901~~ - Add an authentication framework CLOSED
- ~~BOOKKEEPER-959~~ - ClientAuthProvider and BookieAuthProvider Public API used Protobuf Shaded classes RESOLVED
- ~~BOOKKEEPER-391~~ - Support Kerberos authentication of bookkeeper RESOLVED
- ~~BOOKKEEPER-588~~ - SSL support RESOLVED
- ~~BOOKKEEPER-390~~ - Provide support for ZooKeeper authentication RESOLVED