Grouper ActiveMQ integration

ActiveMQ is an open source application messaging middleware service where applications can send messages to other applications (or groups of consumers) in real-time without polling.

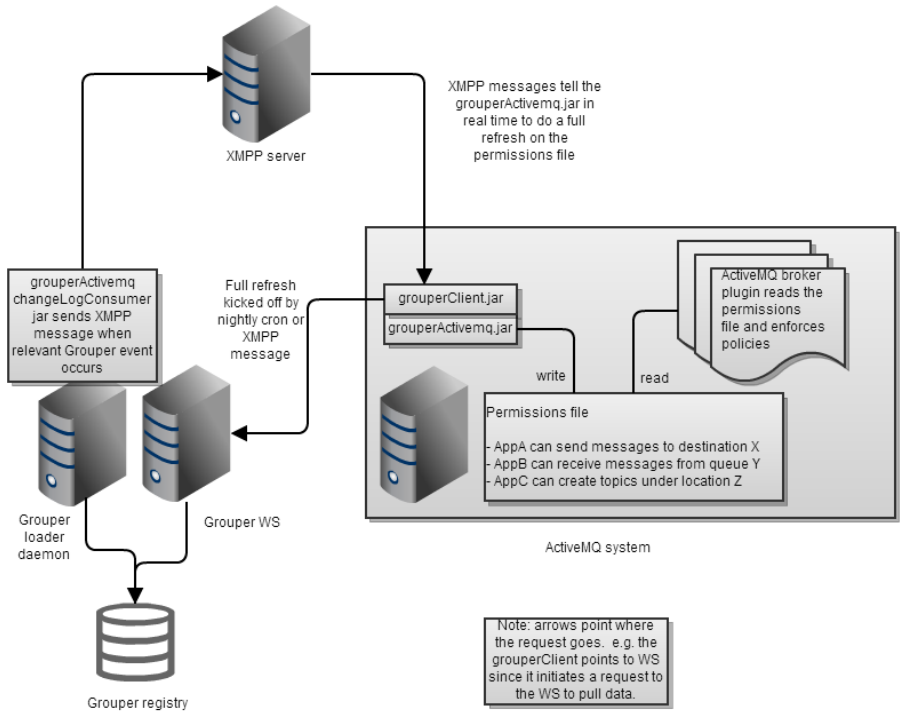
Grouper can be used to manage ActiveMQ authorizations. University of Pennsylvania and Penn State are interested in this component.

ActiveMQ users can send/receive messages to topics/queues, and also can dynamically create topics/queues. Grouper permissions can be exported and kept in sync (cron'ed and real-time) with a local permissions store. An ActiveMQ plugin reads the local permissions and enforces the rules.

The Grouper-ActiveMQ component is a jar that you can add to the ActiveMQ system (along with a few other jars), and some configuration that allows authorization managed from Grouper.

Note: currently when you protect a destination, it is the name of a queue or a topic...

Architecture



Authentication

If you want to authenticate to ActiveMQ with kerberos service principal user/pass, there is a plugin in this Grouper-ActiveMQ component. Configure in the activemq.xml

```
<plugins>
   <!-- ATTENTION! AUTHN MUST BE BELOW AUTHZ!!! -->
   <bean xmlns="http://www.springframework.org/schema/beans"
      id="kerberosPlugin"
      class="edu.internet2.middleware.grouperActivemq.GrouperBrokerPlugin" />

</plugins>
```

Configure kerberos authentication in grouper.activemq.properties:

```
########################################
## KERBEROS settings, only needed if doing kerberos auth
########################################

# number of minute to cache successful logins
kerberos.login.cache.success.minutes = 5

# number of minute to cache failed logins
kerberos.login.cache.failed.minutes = 1
```

Configure the krb5.conf (in classpath)

```
[libdefaults]
    default_realm = UPENN.EDU
    default_tkt_enctypes = des3-hmac-sha1 des-cbc-crc
    default_tgs_enctypes = des3-hmac-sha1 des-cbc-crc

[realms]
    UPENN.EDU = {
        kdc = kerberos1.upenn.edu
        kdc = kerberos2.upenn.edu
        kdc = kerberos3.upenn.edu
        admin_server = kerberos1.upenn.edu
    }
```

Getting started

- This is developeed with Grouper 2.1, and ActiveMQ 5.7
- Check out the Grouper ActiveMQ component: http://anonsvn.internet2.edu/viewvc/viewvc.py/i2mi/branches/GROUPER_2_1_BRANCH/grouper-misc/grouperActivemq/
- Build into a jar with default ant task
- Copy to the ActiveMQ library path: grouperActivemq.jar, grouperClient.jar, smack.jar, quartz.jar, commons-logging.jar, ezmorph.jar, json-lib.jar, jta.jar,
- Copy to the ActiveMQ classpath: grouper.activemq.base.properties, goruper.activemq.properties, grouper.client.example.properties, grouper.client.properties, krb5.conf (if applicable), morphString.example.properties (if applicable), morphString.properties (if applicable), jass.conf (if applicable)
- Configure the Grouper-ActiveMQ connector in the activemq.xml

```
    <plugins>
       <bean xmlns="http://www.springframework.org/schema/beans"
          id="grouperPlugin"
          class="edu.internet2.middleware.grouperActivemq.GrouperBrokerPlugin" />
    </plugins>
```

Configuration

Create a role (or some roles) in Grouper which are used for ActiveMQ users.

Create a permission definition (or definitions if you are delegating) that are used for ActiveMQ permissions.

Make sure all ActiveMQ permission definitions have these actions:

```
/** send a message */
  sendMessage,

  /** send a message on this destination or sub destinations */
  sendMessageInherit,

  /** receive a message */
  receiveMessage,

  /** receive a message on this destination or sub destinations */
  receiveMessageInherit,

  /** create a queue/topic in this destination */
  createDestination,

  /** delete a queue/topic in this destination */
  deleteDestination,

  /** create a queue/topic in this destination or somewhere underneath */
  createDestinationInherit,


  /** create a queue/topic in this destination or somewhere underneath */
  deleteDestinationInherit;
```

Note: you can specify a root folder in Grouper where permissions must be in, and the root folder can be prefix which is removed from the destination name.

Configure the grouper.activemq.properties

```
#
# Grouper activemq configuration
#

# The grouper activemq plugin uses Grouper Configuration Overlays (documented on wiki)
# By default the configuration is read from grouper.activemq.base.properties
# (which should not be edited), and the grouper.activemq.properties overlays
# the base settings.  See the grouper.activemq.base.properties for the possible
# settings that can be applied to the grouper.activemq.properties

########################################
## Config chaining hierarchy
########################################

# comma separated config files that override each other (files on the right override the left)
# each should start with file: or classpath:
# e.g. classpath:grouper.client.example.properties, file:c:/something/myconfig.properties
grouperActivemq.config.hierarchy = classpath:grouper.activemq.base.properties, classpath:grouper.activemq.
properties

# seconds between checking to see if the config files are updated
grouperActivemq.config.secondsBetweenUpdateChecks = 60

########################################
## KERBEROS settings, only needed if doing kerberos auth
########################################

# number of minute to cache successful logins
kerberos.login.cache.success.minutes = 5

# number of minute to cache failed logins
kerberos.login.cache.failed.minutes = 1


########################################
## Permissions settings
########################################
```

```
# prefix for permissions which will be stripped when translating to destination
# if there is nothing specified, then the queue's/topic's are fully qualified in PennGroups
# Note: if you put something in here, you cant easily remove it later...
grouperActivemq.folderPrefix =

# all permissions for activemq must be in this folder
grouperActivemq.requireBaseFolder =

# list all the permission definitions (comma separated) in Grouper where the destinations are specified
# this is a required field.  Note, the actions must exactly match up with GrouperActivemqPermissionAction
grouperActivemq.permissionDefinitions =

# directory on activemq machine which has two cached file for the activemq permissions files:
# activemqGrouperPermissions.1.properties, and activemqGrouperPermissions.2.properties
grouperActivemq.permissionsCacheDirectory =

# the roles that users of activemq must have permissions in, comma separated
grouperActivemq.roleNames =

# subject sources that should be pulled for permissions.  i.e. which subjects can have permissions in
# activemq, and what subject attribute is their login id for activemq.  Increment the number for more entries
# they must be sequential
grouperActivemq.subjectSource.0.sourceId =
grouperActivemq.subjectSource.0.subjectAttributeForLogin =

# grouperActivemq.subjectSource.1.sourceId =
# grouperActivemq.subjectSource.1.subjectAttributeForLogin =

# generally run this every night since real time will run every day
grouperActivemq.fullRefreshQuartzCron = 0 0 7 * * ?
```

Running ActiveMQ from Eclipse

Make a launch with the class: org.apache.activemq.console.Main

The program argument is: start

The VM arguments are:

```
-Dactivemq.home=. -Dactivemq.base=. -Dactivemq.conf=conf -Dactivemq.data=data -Djava.io.tmpdir=temp -XX:
MaxPermSize=130m -Xms200m -Xmx200m
```

Change log consumer

For the change log consumer for XMPP notifications (if you have all permissions coming from one root folder), you can use the same change log consumer (configured differently) as the Unix File permissions consumer:

```
/**
 * @author mchyzer
 * $Id: ClusterLinuxChangeLogConsumer.java,v 1.2 2012/05/21 17:01:24 mchyzer Exp $
 */
package edu.upenn.isc.clusterLinuxClc;

import java.util.List;

import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;

import edu.internet2.middleware.grouper.app.loader.GrouperLoaderConfig;
import edu.internet2.middleware.grouper.changeLog.ChangeLogConsumerBase;
import edu.internet2.middleware.grouper.changeLog.ChangeLogEntry;
import edu.internet2.middleware.grouper.changeLog.ChangeLogLabels;
```

```
import edu.internet2.middleware.grouper.changeLog.ChangeLogProcessorMetadata;
import edu.internet2.middleware.grouper.changeLog.ChangeLogTypeBuiltin;
import edu.internet2.middleware.grouper.util.GrouperUtil;
import edu.internet2.middleware.grouper.xmpp.XmppConnectionBean;


/**
 * change log consumer to listen for specific gruoper actions that should tell linux to refresh
 * their permissions
 */
public class ClusterLinuxChangeLogConsumer extends ChangeLogConsumerBase {

  /** */
  private static final Log LOG = GrouperUtil.getLog(ClusterLinuxChangeLogConsumer.class);

  /**
   * @see edu.internet2.middleware.grouper.changeLog.ChangeLogConsumerBase#processChangeLogEntries(java.util.
List, edu.internet2.middleware.grouper.changeLog.ChangeLogProcessorMetadata)
   */
  @Override
  public long processChangeLogEntries(List<ChangeLogEntry> changeLogEntryList,
      ChangeLogProcessorMetadata changeLogProcessorMetadata) {

    //identifies which config is running, multiple could be running
    String consumerName = changeLogProcessorMetadata.getConsumerName();

    long currentId = -1;

    XmppConnectionBean xmppConnectionBean = null;
    String recipient = null;

    String grouperFolderBase = null;

    {
      String grouperFolderBaseConfigName = "changeLog.consumer." + consumerName + ".clusterLinux.
grouperFolderBase";

      grouperFolderBase = GrouperLoaderConfig.getPropertyString(grouperFolderBaseConfigName, true);
    }

    boolean sendMessage = false;

    for (ChangeLogEntry changeLogEntry : changeLogEntryList) {

      //try catch so we can track that we made some progress
      try {

        currentId = changeLogEntry.getSequenceNumber();

        //only need to send once
        if (sendMessage) {
          continue;
        }

        //this might be a little aggressive, but basically if a permission or member changes in the
clusterLinux folder of
        //grouper then send a refresh message
        //note: not using constants for permissions so it works in 2.0 and 2.1...
        if ((StringUtils.equals("permission", changeLogEntry.getChangeLogType().getChangeLogCategory())
            && (StringUtils.equals("addPermission", changeLogEntry.getChangeLogType().getActionName())
                || StringUtils.equals("deletePermission", changeLogEntry.getChangeLogType().getActionName())))
            || (StringUtils.equals("attributeAssign", changeLogEntry.getChangeLogType().getChangeLogCategory())
                && (StringUtils.equals("addAttributeAssign", changeLogEntry.getChangeLogType().getActionName())
                    || StringUtils.equals("deleteAttributeAssign", changeLogEntry.getChangeLogType().
getActionName())))) {

          String permissionName = changeLogEntry.retrieveValueForLabel("attributeDefNameName");

          if (permissionName != null && permissionName.startsWith(grouperFolderBase)) {
            sendMessage = true;
          }
```

```
          if (LOG.isDebugEnabled()) {
            LOG.debug("Processing changeLog #" + currentId + ", permissionName: " + permissionName
                + ", grouperFolderBase: " + grouperFolderBase
                + ", message: "
                + changeLogEntry.getChangeLogType().getChangeLogCategory() + "."
                + changeLogEntry.getChangeLogType().getActionName() + ", sendMessage: " + sendMessage
                );
          }
        } else if (changeLogEntry.equalsCategoryAndAction(ChangeLogTypeBuiltin.MEMBERSHIP_ADD)
            || changeLogEntry.equalsCategoryAndAction(ChangeLogTypeBuiltin.MEMBERSHIP_DELETE)
            || changeLogEntry.equalsCategoryAndAction(ChangeLogTypeBuiltin.MEMBERSHIP_UPDATE)) {

          String roleName = changeLogEntry.retrieveValueForLabel(ChangeLogLabels.MEMBERSHIP_ADD.groupName);

          if (roleName != null && roleName.startsWith(grouperFolderBase)) {
            sendMessage = true;
          }
          if (LOG.isDebugEnabled()) {
            LOG.debug("Processing changeLog #" + currentId + ", roleName: " + roleName
              + ", grouperFolderBase: " + grouperFolderBase
              + ", message: "
              + changeLogEntry.getChangeLogType().getChangeLogCategory() + "."
              + changeLogEntry.getChangeLogType().getActionName() + ", sendMessage: " + sendMessage
              );
          }

        } else if (StringUtils.equals("permission", changeLogEntry.getChangeLogType().getChangeLogCategory())
            && StringUtils.equals("permissionChangeOnRole", changeLogEntry.getChangeLogType().getActionName()))
{

          //note, this is 2.1+

          String roleName = changeLogEntry.retrieveValueForLabel("roleName");

          if (roleName != null && roleName.startsWith(grouperFolderBase)) {
            sendMessage = true;
          }
          if (LOG.isDebugEnabled()) {
            LOG.debug("Processing changeLog #" + currentId + ", roleName: " + roleName
              + ", grouperFolderBase: " + grouperFolderBase
              + ", message: "
              + changeLogEntry.getChangeLogType().getChangeLogCategory() + "."
              + changeLogEntry.getChangeLogType().getActionName() + ", sendMessage: " + sendMessage
              );
          }
        } else {
          if (LOG.isDebugEnabled()) {
            LOG.debug("Processing changeLog #" + currentId + ", "
              + changeLogEntry.getChangeLogType().getChangeLogCategory() + "."
              + changeLogEntry.getChangeLogType().getActionName() + ", sendMessage: " + sendMessage
              );
          }
        }

        //is there something to send?
        if (sendMessage) {

          String message = "Update permissions";

          if (xmppConnectionBean == null) {

            recipient = GrouperLoaderConfig.getPropertyString("changeLog.consumer."
                + consumerName + ".publisher.recipient", "");

            String xmppServer = GrouperLoaderConfig.getPropertyString("xmpp.server.host");
            int port = GrouperLoaderConfig.getPropertyInt("xmpp.server.port", -1);
            String username = GrouperLoaderConfig.getPropertyString("xmpp.user", "");
            String password = GrouperLoaderConfig.getPropertyString("xmpp.pass", "");
            String resource = GrouperLoaderConfig.getPropertyString("xmpp.resource", "");

            xmppConnectionBean = new XmppConnectionBean(xmppServer, port, username, resource, password);
```

```
          }

          xmppConnectionBean.sendMessage(recipient, message);

        }
      } catch (Exception e) {
        //we unsuccessfully processed this record... decide whether to wait, throw, ignore, log, etc...
        LOG.error("problem with id: " + currentId, e);
        changeLogProcessorMetadata.setHadProblem(true);
        changeLogProcessorMetadata.setRecordException(e);
        changeLogProcessorMetadata.setRecordExceptionSequence(currentId);
        //stop here
        return currentId;
        //continue
      }
    }

    return currentId;
  }

}
```

Configure it like this in the grouper-loader.properties:

```
changeLog.consumer.activemqTest.class = edu.upenn.isc.clusterLinuxClc.ClusterLinuxChangeLogConsumer
changeLog.consumer.activemqTest.publisher.recipient = activemq_recipient@school.edu/activemqTest,
scomuser@school.edu
changeLog.consumer.activemqTest.clusterLinux.grouperFolderBase = test:folder:it_dept:apps:activemq
changeLog.consumer.activemqTest.quartzCron =
```

Local permissions cache

You configure the directory where the permissions files will be stored. There will be two files (in case one doesnt generate fully, there is a spare), here are the sample contents:

```
# automatically generated from Grouper-ActiveMQ connector...

activemq.permission.0.user = fastGrouperAppTest/medley.isc-seo.upenn.edu
activemq.permission.0.permission.0 = sendMessage__testDestination1
activemq.permission.0.permission.1 = receiveMessageInherit__testFolder1:testDestination1_1
activemq.permission.1.user = pennserver_secureshare_test/medley.isc-seo.upenn.edu
activemq.permission.1.permission.0 = sendMessageInherit__testFolder1:testDestination1_1
activemq.permission.1.permission.1 = receiveMessage__testDestination1
activemq.permission.1.permission.2 = createDestination__testDestination1
activemq.permission.1.permission.3 = createDestination__testFolder1:testDestination1_1
activemq.permission.1.permission.4 = deleteDestination__testDestination1

# if this is here, the file is complete
activemq.permissionSuccess = true
```

Broker plugin example

If you have an implementation of Broker:

```
/**
 * @author mchyzer $Id$
 */
package edu.upenn.isc.activemq;

import org.apache.activemq.broker.Broker;
import org.apache.activemq.broker.BrokerFilter;
import org.apache.activemq.broker.ConnectionContext;
```

```
import org.apache.activemq.broker.ProducerBrokerExchange;
import org.apache.activemq.broker.region.Destination;
import org.apache.activemq.broker.region.Subscription;
import org.apache.activemq.command.ActiveMQDestination;
import org.apache.activemq.command.ConsumerInfo;
import org.apache.activemq.command.DestinationInfo;
import org.apache.activemq.command.Message;
import org.apache.activemq.command.ProducerInfo;

/**
 * my broker
 */
public class MyBroker extends BrokerFilter {

  /**
   * @param next1
   */
  public MyBroker(Broker next1) {
    super(next1);

  }

  /**
   * @see org.apache.activemq.broker.BrokerFilter#addDestinationInfo(org.apache.activemq.broker.
ConnectionContext, org.apache.activemq.command.DestinationInfo)
   */
  @Override
  public void addDestinationInfo(ConnectionContext context, DestinationInfo info)
      throws Exception {
    System.out.println("addDestinationInfo: ");
    super.addDestinationInfo(context, info);
  }

  /**
   * @see org.apache.activemq.broker.BrokerFilter#addDestination(org.apache.activemq.broker.ConnectionContext,
org.apache.activemq.command.ActiveMQDestination, boolean)
   */
  @Override
  public Destination addDestination(ConnectionContext context,
      ActiveMQDestination destination, boolean create) throws Exception {
    System.out.println("addDestination: ");
    return super.addDestination(context, destination, create);
  }

  /**
   * @see org.apache.activemq.broker.BrokerFilter#removeDestination(org.apache.activemq.broker.
ConnectionContext, org.apache.activemq.command.ActiveMQDestination, long)
   */
  @Override
  public void removeDestination(ConnectionContext context,
      ActiveMQDestination destination, long timeout) throws Exception {

    System.out.println("removeDestination: ");
    super.removeDestination(context, destination, timeout);
  }

  /**
   * @see org.apache.activemq.broker.BrokerFilter#addConsumer(org.apache.activemq.broker.ConnectionContext, org.
apache.activemq.command.ConsumerInfo)
   */
  @Override
  public Subscription addConsumer(ConnectionContext context, ConsumerInfo info)
      throws Exception {

    System.out.println("addConsumer: ");
    return super.addConsumer(context, info);
  }

  /**
   * @see org.apache.activemq.broker.BrokerFilter#addProducer(org.apache.activemq.broker.ConnectionContext, org.
apache.activemq.command.ProducerInfo)
```

```
   */
  @Override
  public void addProducer(ConnectionContext context, ProducerInfo info) throws Exception {

    System.out.println("addProducer: ");
    super.addProducer(context, info);
  }

  /**
   *
   * @see org.apache.activemq.broker.BrokerFilter#send(org.apache.activemq.broker.ProducerBrokerExchange, org.
apache.activemq.command.Message)
   */
  @Override
  public void send(ProducerBrokerExchange producerExchange, Message messageSend)
      throws Exception {
    System.out.println("send: ");
    super.send(producerExchange, messageSend);
  }

}
```

And you have an implementation for BrokerPlugin:

```
/**
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package edu.upenn.isc.activemq;

import org.apache.activemq.broker.Broker;
import org.apache.activemq.broker.BrokerPlugin;

/**
 * An authorization plugin where each operation on a destination is checked
 * against an authorizationMap
 *
 * @org.apache.xbean.XBean
 *
 *
 */
public class MyBrokerPlugin implements BrokerPlugin {

    public MyBrokerPlugin() {
    }

    public Broker installPlugin(Broker broker) {
        return new MyBroker(broker);
    }

}
```

Then you can register the broker in the activemq.xml (add above <transportConnectors>

```
    <plugins>
       <bean xmlns="http://www.springframework.org/schema/beans"
          id="myBrokerPlugin"
          />

    </plugins>
```

sdf