SimpleAuthenticator

Cassandra uses a simple and pluggable authentication and authorization model using the supplied interfaces:

```
org.apache.cassandra.auth.IAuthenticator.java
org.apache.cassandra.auth.IAuthority.java
```

Out of the box Cassandra ships with a promiscuous implementation that allows all access to all users without the need to log in. If you want to increase the security beyond this you are free to implement the above interfaces to provide whatever security mechanisms you like.

In the source distribution there is a simple example of authentication and authorization based on entries in properties files. This implementation is not provided with the binary distribution as it may not provide a level of security that is adequate for your needs. However, you can use these examples for your own implementations. This example is found in the examples directory, in the package

```
org.apache.cassandra.auth.SimpleAuthenticator.java
```

To use this implementation, move the classes to the cassandra source tree, and recompile the jars with their inclusion. Then move the example property files

```
access.properties
passwd.properties
```

to your conf directory. You can refer to these files for the format needed by the SimpleAuthenticator class.

You enable it by adding

```
authenticator: org.apache.cassandra.auth.SimpleAuthenticator
authority: org.apache.cassandra.auth.SimpleAuthority
```

in conf/cassandra.yaml If no authenticator is specified, the default is org.apache.cassandra.auth.AllowAllAuthenticator

If you use SimpleAuthenticator you should also update bin/cassandra.in.sh to specify additional properties which point to the location of your authentication files. Add the following to the JVM_OPTS (e.g. before the -Dcom.sun.management.jmxremote.port setting):

```
        -Dpasswd.properties=/usr/local/apache-cassandra-1.0.0/conf/passwd.properties \
        -Daccess.properties=/usr/local/apache-cassandra-1.0.0/conf/access.properties \
```

(Alter the paths to the configuration files depending on where placed the files.)

Caveats

Note that this authentication/authorization is applied to Thrift requests from clients. It is not applied to inter-node messages. This means that an attacker with access to the network used by your cluster could bypass authentication/authorization to cause damage or extract data, by directly crafting and sending inter-node messages. Server nodes should therefore be protected from clients and other hosts by firewall rules.

If there are untrusted people or systems on your datacenter network, you can additionally enable inter-node encryption, which prevents the type of attack described in the previous paragraph.