Basic Hints on Security Configuration

Where is the security configuration?

In a normal Geronimo server, the basic security configuration is divided into two plugins, **j2ee-security** and **server-security-config**. The parts you are not too likely to want to change, such as the jacc provider and the keystore manager, are in j2ee-security. The parts that you are almost certain to want to change is in server-security config. For instance, the toy properties file security realm for the admin console is in server-security-config.

So I have an enterprise wide authentication system.... how do I set it up for all my apps?

You want to replace server-security-config with your own Geronimo plugin (see Administering plugins) that contains a security realm customized for your security setup (e.g. ldap) and includes whatever keystores you need. To replace all uses of server-security-config with your plugin, include an artifact-alias element in your `geronimo-plugin.xml` file.

```
<artifact-alias key="org.apache.geronimo.framework/server-security-config/2.2-SNAPSHOT/car">com.myco/myco-
security-config/1.0/car</artifact-alias>
<artifact-alias key="org.apache.geronimo.framework/server-security-config//car">com.myco/myco-security-config/1.
0/car</artifact-alias>
```

Another option is to use maven with the car-maven-plugin. The above code would need to be included in the car-maven-plugin configuration in `pom.xml`.

Note that if you want the admin console and MEJB to continue working without redeployment, you have to include a security realm named *geronimo-admin*. geronimo-admin should supply supply appropriate users with principals of class org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal and names of (as appropriate) admin (for console and MEJB read access) and mejbadmin (for MEJB write access).

As with any geronimo plugin, you can include any jars in the plugin's classloader by installing the jars in the geronimo repository and listing them as dependencies in the geronimo plan. The car-maven-plugin can be used to make the geronimo dependencies the same as the maven dependencies and to have plugin installation also install all the needed jars.

I'm still doing experiments and am not ready to write a plugin... how do I use a realm I created in the admin console?

While getting all your configuration into plugins with source code in scm and built by maven provides a completely reproducible environment, you might want to experiment with a security realm you set up using the admin console. In this case you need to, while geronimo is stopped, edit the `var/config/artifact-aliases.properties` file by hand. Assuming that you named the configuration *geronimo-admin* the console will come up with a plugin id of console.realm/geronimo-admin/1.0/car. You need to put lines like:

```
org.apache.geronimo.framework/server-security-config/2.2-SNAPSHOT/car=console.realm/geronimo-admin/1.0/car
org.apache.geronimo.framework/server-security-config//car=console.realm/geronimo-admin/1.0/car
```

where you've replaced *2.2-SNAPSHOT* with the actual version of geronimo you are using.

Who needs enterprise-wide? I want my app to include its own security setup!

You can also include security realm configuration, keystores, and credential stores in your geronimo plan for your application. Just put the gbean configurations at the end after the javaee specific configuration. In this case you may not want to remove the standard server-security-config as removing it would prevent the admin console or mejb from starting.

For Web applications using Spring Security

Spring security may secure spring applications but it won't relate to container managed authorization in Geronimo unless you do something to hook it up. You need some code that looks vaguely like this:

```
Subject subject = getSpringAuthenticatedSubject();
ContextManager.registerSubject(subject); //if the subject is cached in a session this should only happen once
when the subject is first authenticated/constructed.

//the following should happen on every request
ContextManager.setCallers(subject, subject);
try {
//process request
} finally {
   ContextManager.clearCallers();
}
```

Using a pluggable encryption system

By default you get the old behavior with "{Simple}" encryption with a hard-coded key. If you want to have a fixed key generated by Geronimo, you can add this Gbean to the rmi-naming module in `config.xml`:

```
<gbean name="org.apache.geronimo.configs/rmi-naming/2.2-SNAPSHOT/car?name=ConfiguredEncryption,j2eeType=GBean"
gbeanInfo="org.apache.geronimo.system.util.ConfiguredEncryption">
<attribute name="path">var/security/ConfiguredSecretKey.ser</attribute>
<reference name="ServerInfo"><pattern><name>ServerInfo</name></pattern></reference>
</gbean>
```

This will create a key the first time the server started, after that it will keep using the saved key at the location specified. If you put a serialized SecretKeySpec there it will use it instead.