Configuring RBAC in JBoss EAP and Wildfly - Part One

by Andy Overton · Dec. 09, 14 · Interview

Like (0) Comment (0) Save Tweet Share 8.37K Views

In this blog post I will look into the basics of configuring Role Based Access Control (RBAC) in EAP and Wildfly.

RBAC was introduced in EAP 6.2 and WildFly 8 so you will need either of those if you wish to use RBAC.

For the purposes of this blog I will be using the following:

OS - Ubuntu 14
Java - 1.7.0_67
JBoss - EAP 6.3

Although I'm using EAP these instructions should work just the same on Wildfly.

What is RBAC?

Role Based Access Control is designed to restrict system access by specifying permissions for management users. Each user with management access is given a role and that role defines what they can and cannot access.
In EAP 6.2+ and Wildfly 8+ there are seven predefined roles each of which has different permissions. Details on each of the roles can be found here:
https://access.redhat.com/documentation/en-US/JBoss_Enterprise_Application_Platform/6.2/html/Security_Guide/Supported_Roles.html
In order to authenticate users one of the three standard authentication providers must be used. These are:
Local User - The local user is automatically added as a SuperUser so a user on the server machine has full access. This user should be removed in a production system and access locked down to named users.
Username/Password - using either the mgmt-users.properties file, or an LDAP server.
Client Certificate - using a trust store
For the purposes of this blog and to keep things simple we will use username/passwords and the mgmt-users.properties file

Why do we need RBAC?

The easiest way to show this is through a practical demo.

Configuration can be done either via the Management Console or via the Command Line Interface (CLI). However, only a limited set of tasks can be done via the management console whereas all tasks are available via the CLI. Therefore, for the purposes of this blog I will be doing all configuration via the CLI.

In our test scenario we have 4 users:

Andy - This user is the main sys-admin and therefore we want him to be able to access everything.

Bob - This user is a lead developer and therefore will need to be able to deploy apps and make changes to certain application resources.

Clare & Dave - These users are standard developers and will need to be able to view application resources but should not be able to make changes.

First of all we will set up a number of users.

In order to do so we will use the add-user.sh script which can be found in:

```
<JBOSS_INSTALL_DIR>/bin
```

Create the following users in the stated groups. (Enter No for the final question for all users)

Andy - no group

Bob - lead-developers

Clare - standard-developers

Dave - standard-developers

In <JBOSS_INSTALL_DIR>/domain/configuration you will find a file called mgmt-users.properties.

At the bottom of this file you will see a list of the users we've created similar to this:

Andy=82153e0297590cceb14e7620ccd3b6ed

Bob=06a61e836d9d2d5be98517b468ab72cc

Clare=63a8ff615a122c56b1d47fc098ff5124

Dave=2df8d1e02e7f3d13dcea7f4b022d0165

In the same directory you will find a a file called mgmt-groups.properties, at the bottom of this file you will see a list of users and the groups they are in, like so:

Andy=

Bob=lead-developers

Clare=developers

Dave=developers

Now point a browser at http://localhost:9990 and log in as the user Dave. Navigate around and you will see you have full access to everything.

This is precisely why RBAC is needed! Allowing all users to not only access the management console but to be able to access and alter anything is a recipe for disaster and guaranteed to cause issues further down the line. Often users don't understand the implications of the changes they have made, it may just be a quick fix to resolve an immediate issue but it may have long term consequences that are not noticed until much further down the line when the changes that were made have been forgotten about or are not documented. As someone who works in support we see these kind of issues on a regular basis and they can be difficult to track down with no audit trail and users not realising that the minor change they made to one part of the system is now causing a major issue in some other part of the system.

OK, so we now have our users set up but at the moment they have full access to everything. Next up we will configure these users and assign them to roles.

First of all start up the CLI.

Run the following command:

```
<JBOSS_INSTALL_DIR>/bin/jboss-cli.sh -c
```

Change directory to the authorisation node

```
cd /core-service=management/access=authorization
```

Running the following command lists the current role names and the standard role names along with two other attributes

```
ls -l
```

The two we are interested in here are permission-combination-policy and provider.

The permission-combination-policy defines how permissions are determined if a user is assigned more than one role. The default setting is permissive. This means that if a user is assigned to any role that allows a particular action then the user can perform that action.

The opposite of this is rejecting. This means that if a user is assigned to multiple roles then all those roles must permit an action for a user to be able to perform that action.

The other attribute of interest here is provider. This can be set to either simple (which is the default) or rbac. In simple mode all management users can access everything and make changes, as we have seen. In rbac mode users are assigned roles and each of those roles has difference privileges.

Switching on RBAC

OK, lets turn on RBAC...

Run the following commands to turn on RBAC

```
cd /core-service=management/access=authorization

:write-attribute(name=provider, value=rbac)
```

Restart JBoss

Now point a browser at http://localhost:9990 and try to log in as the user Andy (who should be able to access everything).

You should see the following message :

Insufficient privileges to access this interface.

This is because at the moment the user Andy isn't mapped to any role. Let's fix that now:

If you look in domain.xml in the management element you will see the following:

```
<access-control provider="rbac">
    <role-mapping>
        <role name="SuperUser">
            <include>
                <user name="$local"/>
            </include>
        </role>
    </role-mapping>
</access-control>
```

This shows that at the moment only the local user is mapped to the SuperUser role.

Mapping users and groups to roles

We need to map our users to the relevant roles to allow them access.
In order to do this we need the following command:

```
role-mapping=ROLENAME/include=ALIAS:add(name=USERNAME, type=USER)
```

Where rolename is one of the pre-configured roles, alias is a unique name for the mapping and user is the name of the user to map.
So, lets map the user Andy to the SuperUser role.

```
./role-mapping=SuperUser/include=user-Andy:add(name=Andy, type=USER)
```

In domain.xml you will see that our user has been added to the SuperUser role:

```
<access-control provider="rbac">
    <role-mapping>
        <role name="SuperUser">
            <include>
                <user name="$local"/>
                <user name="Andy"/>
            </include>
        </role>
    </role-mapping>
</access-control>
```

Now point a browser at http://localhost:9990 you should now be able to log in as the user Andy and have full access to everything.

Next we need to add mappings for the other roles we want to use.

```
./role-mapping=Deployer:add
./role-mapping=Monitor:add
```

Now we need to give role mappings to all our other users. As we have them in groups we can assign the groups to roles, rather than mapping by user.

The command is basically the same as for a user but the type is GROUP rather than user.

Here we are mapping lead developers to the Deployer role and standard developers to the Monitor role.

```
./role-mapping=Deployer/include=group-lead-devs:add(name=lead-developers, type=GROUP)
./role-mapping=Monitor/include=group-standard-devs:add(name=developers, type=GROUP)
```

If you look in domain.xml you should now see the following showing that the user Andy is mapped to the SuperUser role and the two groups are mapped to the Deployer and Monitor roles.

```
<management>
    <access-control provider="rbac">
        <role-mapping>
            <role name="SuperUser">
                <include>
                    <user name="$local"/>
                    <user name="Andy"/>
                </include>
            </role>
            <role name="Deployer">
                <include>
                    <group alias="group-lead-devs" name="lead-developers"/>
                </include>
            </role>
            <role name="Monitor">
                <include>
                    <group alias="group-standard-devs" name="developers"/>
                </include>
            </role>
        </role-mapping>
    </access-control>
</management>
```

You can also view the role mappings in the admin console.

Click on the Administration tab.

Expand the Access Control item on the left and select Role Assignment.

Select the Users tab - this shows users that are mapped to roles.

Select the Groups tab and you will see the mapping between groups and roles.

Log in as the different users and see the differences between what you can and can't access.

Conclusion

So, that's it for Part One. We have switched on RBAC, set up a number of users and groups and mapped those users and groups to particular roles to give them different levels of access.

In Part Two of this blog I will look at constraints which allow more fine grained permission setting, scoped roles which allow you to set permissions on individual servers and audit logging which allows you to see who is accessing the management console and see what changes they are making.