Access Control: Authentication & Authorization

Access Control Mechanisms

- Declarative using Security Realms
  - Separate from your jsp and servlet code
  - Based on specifying centralized policy
  - Based on static roles who are groups of users that have access to particular resources (typically pages)

- Programmatic
  - Your code is responsible
  - Choose when you need to create intricate access control strategies

Security Using Realms

- PLUS: Really simple!
- MINUS: Static policy – very rarely a problem
- File-based (MemoryRealm), JDBC, JNDI are "ready out of the box"
- MemoryRealm (tomcat-users.xml) has extra downside of clear text passwords
An example of declarative authorization w/ tomcat-users.xml

- Three classes of users
  - chair, committee, admin
- All http://host/app/admin/* pages will be accessed only by administrators
- All http://host/app/chair/* pages will be accessed only by the chairs
- All http://host/app/committee/* pages will be accessed only by chairs and committee
- "abk" is committee member
- "vianu" is chair
- "akovacev" is administrator

Security constraints (part of web.xml)

```xml
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Admin Area</web-resource-name>
        <url-pattern>/admin/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>admin</role-name>
    </auth-constraint>
    <user-data-constraint>
        <description>SSL Required</description>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>

<security-constraint>
    <web-resource-collection>
        <web-resource-name>Committee Area</web-resource-name>
        <url-pattern>/committee/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>chair</role-name>
        <role-name>committee</role-name>
    </auth-constraint>
    <user-data-constraint>
        <description>SSL Required</description>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Security Constraints

```xml
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Committee Area</web-resource-name>
        <url-pattern>/committee/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>chair</role-name>
    </auth-constraint>
    <user-data-constraint>
        <description>SSL Required</description>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```
Dynamic Database-Driven Access Control

- **Tomcat-users.xml** is a kind of *Security Realm*, i.e., a provider of user credentials
- **JDBCRealm**: User credentials are stored in a database, accessed via JDBC
- **JNDIRealm**: User credentials are stored in directory server, accessed via JNDI
Realm in `server.xml`

```xml
<Realm className="org.apache.catalina.realm.JDBCRealm" debug="99"
driverName="org.gjt.mm.mysql.Driver"
connectionURL="jdbc:mysql://localhost/authority"
connectionName="test"
digest="MD5"
userTable="users"
userNameCol="username"
userCredCol="password"
userRoleTable="userroles"
roleNameCol="rolename" />
```

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>username</strong></td>
<td><strong>password</strong></td>
<td></td>
</tr>
<tr>
<td>vianu</td>
<td>foo</td>
<td></td>
</tr>
<tr>
<td>abk</td>
<td>foo</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>username</strong></td>
<td><strong>rolename</strong></td>
<td></td>
</tr>
<tr>
<td>vianu</td>
<td>committee</td>
<td></td>
</tr>
<tr>
<td>abk</td>
<td>chair</td>
<td></td>
</tr>
</tbody>
</table>

Scope of Realm

- Depending on which level of the `server.xml` you place declaration
- If at Engine level it applies to all apps
- If at Context level only on the enclosing app

Encryption

- Notice routing to 8443
- Key exchanged in first step of session is used for encryption of subsequent communications
Hiding Passwords

// Assume pwd has password and user has username
// and con is connection to database of JDBCRealm used
// for security

String encMD5Pwd = RealmBase.Digest(pwd, "MD5"); // returns MD5 encoding
// which you insert in DB

PreparedStatement makeNewUser = con.prepareStatement("INSERT INTO users (username, password) values (?, ?)");
makeNewUser.setString(1, user);
makeNewUser.setString(2, encMD5Pwd);
makeNewUser.execute();