spring-security-ldap - Reference Documentation

Authors: Burt Beckwith

Version: 0.1.1

Table of Contents

1. Introduction

The LDAP plugin adds support for LDAP and Active Directory authentication to a Grails application that uses Spring Security. It depends on the Spring Security Core plugin.
Once you have configured your Grails application as an LDAP client you can delegate authentication to LDAP and not have to manage users' authentication information in your application. By default roles are inferred from LDAP group membership and you can store additional application-specific roles in your database.
Please refer to the Spring Security LDAP documentation for details of the underlying implementation.

1.1 History

- Version 0.1
  - released 06/18/2010

2. Usage

Configuring your LDAP server is beyond the scope of this document. There are many different approaches and this will most likely be done by IT staff. It's assumed here that you already have a running LDAP or Active Directory server.

There isn't much that you need to do in your application to use LDAP. Just install this plugin, and configure any required parameters and whatever optional parameters you want in Config.groovy. These are described in detail in Chapter 3 but typically you only need to set these properties

```
grails.plugins.springsecurity.ldap.context.managerDn = 'uid=admin,ou=system'
grails.plugins.springsecurity.ldap.context.managerPassword = 'secret'
grails.plugins.springsecurity.ldap.context.server = 'ldap://localhost:10389'
grails.plugins.springsecurity.ldap.authorities.groupSearchBase =
     'ou=groups,dc=yourcompany,dc=com'
grails.plugins.springsecurity.ldap.search.base = 'dc=yourcompany,dc=com'
```

Often all role information will be stored in LDAP, but if you want to also assign application-specific roles to users in the database, then add this

```
grails.plugins.springsecurity.ldap.authorities.retrieveDatabaseRoles = true
```

to do an extra database lookup after the LDAP lookup.
Depending on how passwords are encrypted in LDAP you may also need to configure the encryption algorithm, e.g.

```
grails.plugins.springsecurity.password.algorithm = 'SHA-256'
```

Custom UserDetailsContextMapper
There are three options for mapping LDAP attributes to `UserDetails` data (as specified by the `grails.plugins.springsecurity.ldap.mapper.userDetailsClass` config attribute) and hopefully one of those will be sufficient for your needs. If not, it's easy to implement UserDetailsContextMapper yourself.
Create a class in `src/groovy` or `src/java` that implements UserDetailsContextMapper and register it in `grails-app/conf/spring/resources.groovy`:

```
import com.mycompany.myapp.MyUserDetailsContextMapper
beans = {
   ldapUserDetailsMapper(MyUserDetailsContextMapper) {
      // bean attributes
   }
}
```

3. Configuration

Any property overrides must be specified in `grails-app/conf/Config.groovy` using the `grails.plugins.springsecurity` suffix, for example

```
grails.plugins.springsecurity.ldap.search.searchSubtree = true
```

There are several configuration options for the LDAP plugin. In practice the defaults are fine and only a few will need to be overridden.

Name	Default	Meaning
ldap.search.searchSubtree	`true`	If `true` then searches the entire subtree as identified by context, if false (the default) then only searches the level identified by the context.
ldap.search.base	''	Context name to search in, relative to the base of the configured ContextSource, e.g. 'dc=example,dc=com', 'ou=users,dc=example,dc=com'
ldap.search.filter	`'(uid={0})'`	The filter expression used in the user search
ldap.search.derefLink	`false`	Enables/disables link dereferencing during the search
ldap.search.timeLimit	`0` (unlimited)	The time to wait before the search fails
ldap.search. attributesToReturn	`null` (all)	The attributes to return as part of the search
ldap.authenticator.useBind	`true`	if `true` uses a `BindAuthenticator` to bind as the authenticating user, if `false` uses a `PasswordComparisonAuthenticator` to lookup the user login name and compare passwords
ldap.authenticator. attributesToReturn	`null` (all)	names of attribute ids to return; use `null` to return all and an empty list to return none
ldap.authenticator.dnPatterns	`null` (none)	optional pattern(s) used to create DN search patterns, e.g. ["cn={0},ou=people"]
ldap.authenticator. passwordAttributeName	`'userPassword'`	the name of the password attribute to use when `useBind` = `false`
ldap.mapper. convertToUpperCase	`true`	whether to uppercase retrieved role names (will also be prefixed with "ROLE_")
ldap.mapper. passwordAttributeName	`'userPassword'`	password attribute name to use when building the `UserDetails`
ldap.mapper. userDetailsClass	`null` (create an LdapUserDetailsImpl)	use `'person'` to create a `Person`, `'inetOrgPerson'` to create an `InetOrgPerson`, or `null` to create an `LdapUserDetailsImpl`
ldap.mapper.roleAttributes	`null`	optional names of role attributes
ldap.auth. hideUserNotFoundExceptions	`true`	if `true` throw a new `BadCredentialsException`, otherwise throw the original `UsernameNotFoundException`

ldap.auth.useAuthPassword	`true`	If `true` use the supplied password as the credentials in the authenticationtoken, otherwise obtain the password from the UserDetails object (it may not be possible to read the password from the directory)
ldap.context.managerDn	`'cn=admin,dc=example,dc=com'`	DN to authenticate with
ldap.context.managerPassword	`'secret'`	username to authenticate with
ldap.context.server	`'ldap://localhost:389'`	address of the LDAP server
ldap.context.contextFactoryClassName	`com.sun.jndi.ldap.LdapCtxFactory`	class name of the `InitialContextFactory` to use
ldap.context.dirObjectFactoryClassName	DefaultDirObjectFactory	class name of the DirObjectFactory to use
ldap.context.baseEnvironmentProperties	none	extra context properties
ldap.context.cacheEnvironmentProperties	`true`	whether environment properties should be cached between requsts
ldap.context.anonymousReadOnly	`false`	whether an anonymous environment should be used for read-only operations
ldap.context.referral	`null` ('ignore')	the method to handle referrals. Can be `'ignore'` or `'follow'` to enable referrals to be automatically followed
ldap.authorities.retrieveGroupRoles	`true`	whether to infer roles based on group membership
ldap.authorities.retrieveDatabaseRoles	`false`	whether to retrieve additional roles from the database using the User/Role many-to-many
ldap.authorities.groupRoleAttribute	`'cn'`	The ID of the attribute which contains the role name for a group
ldap.authorities.groupSearchFilter	`'uniquemember={0}'`	The pattern to be used for the user search. {0} is the user's DN
ldap.authorities.searchSubtree	`true`	If `true` a subtree scope search will be performed, otherwise a single-level search is used
ldap.authorities.groupSearchBase	`'ou=groups,dc=example,dc=com'`	The base DN from which the search for group membership should be performed
ldap.authorities.ignorePartialResultException	`false`	Whether `PartialResultException`s should be ignored in searches, typically used with Active Directory since AD servers often have a problem with referrals.
ldap.authorities.defaultRole	none	An optional default role to be assigned to all users