Spring Security LDAP Plugin - Reference Documentation

Burt Beckwith

Version 3.0.0

Table of Contents

1. Introduction to the Spring Security LDAP Plugin

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. Release History

- December 8, 2015
  - 3.0.0 release
- December 7, 2015
  - 2.0.0 release
- November 16, 2015
  - 3.0.0.M1 release
- October 5, 2013
  - 2.0-RC2 release
  - JIRA Issues
- May 22, 2012
  - 1.0.6 release
  - JIRA Issues
- July 31, 2011
  - 1.0.5 release
  - JIRA Issues
- April 19, 2011
  - 1.0.4 release
  - JIRA Issues
- March 12, 2011
  - 1.0.3 release

    - JIRA Issues
- February 14, 2011
    - 1.0.2 release
    - JIRA Issues
- August 01, 2010
    - 1.0.1 release
- July 27, 2010
    - 1.0 release
    - JIRA Issues
- June 18, 2010
    - 0.1 release

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.

2.1. Installation

There isn't much that you need to do in your application to use LDAP. Add a dependency in `build.gradle` for this plugin:

```
dependencies {
   ...
   compile 'org.grails.plugins:spring-security-ldap:3.0.0'
}
```

then configure any required parameters and whatever optional parameters you want in application.yml or application.groovy. These are described in detail in the Configuration section but typically you only need to set these properties:

```
grails:
   plugin:
      springsecurity:
         ldap:
            context:
               managerDn: 'uid=admin,ou=system'
               managerPassword: secret
               server: 'ldap://localhost:10389'
            authorities:
               groupSearchBase: 'ou=groups,dc=yourcompany,dc=com'
            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:
   plugin:
      springsecurity:
         ldap:
            authorities:
               retrieveDatabaseRoles: true
```

to do an extra database lookup after the LDAP lookup.

Depending on how passwords are hashed in LDAP you may also need to configure the hash algorithm, e.g.

```
grails:
   plugin:
      springsecurity:
         password:
            algorithm: 'SHA-256'
```

2.2. Sample config settings for Active Directory

Active directory is somewhat different although still relatively painless if you know what you are doing. Use these example configuration options to get started (tested in Windows Server 2008):

Replace the placeholders inside [] brackets with appropriate values and remove the [] chars

```
grails:
   plugin:
      springsecurity:

         # LDAP config
         providerNames: ['ldapAuthProvider', 'anonymousAuthenticationProvider'] # specify
this when you want to skip attempting to load from db and only use LDAP
         ldap:
            context:
               managerDn: '[distinguishedName]'
               managerDn: '[distinguishedName]'
               managerPassword: '[password]'
               server: 'ldap://[ip]:[port]/'
            authorities:
               ignorePartialResultException: true # typically needed for Active Directory
            search:
               base: '[the base directory to start the search.  usually something like
dc=mycompany,dc=com]'
               filter: 'sAMAccountName={0}' # for Active Directory you need this
               searchSubtree: true
               attributesToReturn: ['mail', 'displayName'] # extra attributes you want
returned; see below for custom classes that access this data
            auth:
               hideUserNotFoundExceptions: false

            # role-specific LDAP config
            useRememberMe: false
            authorities:
               retrieveGroupRoles: true
               groupSearchBase: '[the base directory to start the search.  usually
something like dc=mycompany,dc=com]'
               # If you don't want to support group membership recursion (groups in
groups),
               # then use the following setting
               # grails.plugin.springsecurity.ldap.authorities.groupSearchFilter =
'member={0}'
               # If you wish to support groups with group as members (recursive groups),
use
               # the following:
               # groupSearchFilter: '(member:1.2.840.113556.1.4.1941:={0})'
```

2.3. Custom UserDetailsContextMapper

There are three options for mapping LDAP attributes to `UserDetails` data (as specified by the `grails.plugin.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 Groovy or Java class in `src/main/groovy` that implements UserDetailsContextMapper and register it in `grails-app/conf/spring/resources.groovy`:

```
import com.mycompany.myapp.MyUserDetailsContextMapper

beans = {
   ldapUserDetailsMapper(MyUserDetailsContextMapper) {
      // bean attributes
   }
}
```

For example, here's a custom `UserDetailsContextMapper` that extracts three additional fields from LDAP (fullname, email, and title)

```
package com.mycompany.myapp

import org.springframework.ldap.core.DirContextAdapter
import org.springframework.ldap.core.DirContextOperations
import org.springframework.security.core.userdetails.UserDetails
import org.springframework.security.ldap.userdetails.UserDetailsContextMapper

class MyUserDetailsContextMapper implements UserDetailsContextMapper {

   UserDetails mapUserFromContext(DirContextOperations ctx, String username,
                                  Collection authorities) {

      String fullname = ctx.originalAttrs.attrs['name'].values[0]
      String email = ctx.originalAttrs.attrs['mail'].values[0].toString().toLowerCase()
      def title = ctx.originalAttrs.attrs['title']

      new MyUserDetails(username, null, true, true, true, true,
                        authorities, fullname, email,
                        title == null ? '' : title.values[0])
   }

   void mapUserToContext(UserDetails user, DirContextAdapter ctx) {
      throw new IllegalStateException("Only retrieving data from AD is currently
supported")
   }
}
```

and a custom `UserDetails` class to hold the extra fields:

```
package com.mycompany.myapp

import org.springframework.security.core.GrantedAuthority
import org.springframework.security.core.userdetails.User

class MyUserDetails extends User {

   // extra instance variables
   final String fullname
   final String email
   final String title

   MyUserDetails(String username, String password, boolean enabled, boolean
accountNonExpired,
         boolean credentialsNonExpired, boolean accountNonLocked,
         Collection<GrantedAuthority> authorities, String fullname,
         String email, String title) {

      super(username, password, enabled, accountNonExpired, credentialsNonExpired,
            accountNonLocked, authorities)

      this.fullname = fullname
      this.email = email
      this.title = title
   }
}
```

Here we extend the standard Spring Security `User` class for convenience, but you could also directly implement the interface or use a different base class.

3. Configuration

Any property overrides must be specified in `grails-app/conf/application.yml` (or `application.groovy`) using the `grails.plugin.springsecurity` suffix, for example

```
grails:
   plugin:
      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 `PasswordComparison Authenticator` 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_" or the overridden role prefix)

Name	Default	Meaning
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. hideUserNotFound Exceptions	`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

Name	Default	Meaning
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 GORM
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. ignorePartialResult Exception	`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
ldap.authorities.prefix	`'ROLE_'`	The prefix prepended to group names in order to make them Spring Security Roles.
ldap.authorities.clean.prefix	none	An optional string prefix to strip from the beginning of LDAP group names. For example, `'EnHS-'` will change `EnHS-Staff-All` to `ROLE_Staff-All`
ldap.authorities.clean.suffix	none	An optional string suffix to strip from the end of LDAP group names. For example, `'Group'` will change `Faculty Group` to `ROLE_Faculty`

Name	Default	Meaning
ldap.authorities.clean.dashes	`false`	Set this to true to replace all dashes with underscores in LDAP group names. For example, `Staff-All` will become `ROLE_Staff_All`
ldap.authorities.clean.uppercase	`false`	Set this to true to uppercase all LDAP group names. For example, `My_Ldap_Group` will become `ROLE_MY_LDAP_GROUP`

3.1. Persistent Logins

To use cookies for persistent logins, configure these properties:

Just like with non-LDAP persistent tokens, you need to run the `s2-create-persistent-token` script to create a persistent login domain class and enable the feature.

Name	Default	Meaning
ldap.useRememberMe	`false`	Whether to use persistent logins
ldap.rememberMe.detailsManager. attributesToRetrieve	`null` (all)	The attributes to return as part of the search
ldap.rememberMe.detailsManager. groupMemberAttributeName	'uniquemember'	The attribute which contains members of a group
ldap.rememberMe.detailsManager. groupRoleAttributeName	'cn'	The attribute which corresponds to the role name of a group
ldap.rememberMe.detailsManager. groupSearchBase	'ou=groups,dc=example,dc=com'	The DN under which groups are stored
ldap.rememberMe.detailsManager. passwordAttributeName	'userPassword'	Password attribute name
ldap.rememberMe.usernameMapper. userDnBase	none, must be set, e.g. 'dc=example,dc=com', 'ou=users,dc=example,dc=com'	The base name of the DN
ldap.rememberMe.usernameMapper. usernameAttribute	none, must be set, e.g. 'cn'	the attribute to append for the username component