

LDAP authentication support for the Spring Security plugin.

Spring Security LDAP Plugin - Reference Documentation

Authors: Burt Beckwith

Version: 2.0.0

Table of Contents

- 1** Introduction to the Spring Security LDAP Plugin
 - 1.1** History
- 2** Usage
- 3** Configuration

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 History

- December 7, 2015
 - 2.0.0 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.

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 guide:3. Configuration but typically you only need to set these properties

```
grails.plugin.springsecurity.ldap.context.managerDn = 'uid=admin,ou=system'
grails.plugin.springsecurity.ldap.context.managerPassword = 'secret'
grails.plugin.springsecurity.ldap.context.server = 'ldap://localhost:10389'
grails.plugin.springsecurity.ldap.authorities.groupSearchBase =
'ou=groups,dc=yourcompany,dc=com'
grails.plugin.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.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 hashing algorithm, e.g.

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

Sample Config.groovy 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

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

// role-specific LDAP config
grails.plugin.springsecurity.ldap.useRememberMe = false
grails.plugin.springsecurity.ldap.authorities.retrieveGroupRoles = true
grails.plugin.springsecurity.ldap.authorities.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}' // Active Directory specific
// If you wish to support groups with group as members (recursive groups), use
the following
grails.plugin.springsecurity.ldap.authorities.groupSearchFilter =
'(member:1.2.840.113556.1.4.1941:={0})' // Active Directory specific
```

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 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
    }
}
```

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/Config.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
<code>ldap.search.searchSubtree</code>	<code>true</code>	If <code>true</code> then searches the entire subtree as identified by context, if <code>false</code> (the default) then only searches the level identified by the context.
<code>ldap.search.base</code>	<code>"</code>	Context name to search in, relative to the base of the configured <code>ContextSource</code> , e.g. <code>'dc=example,dc=com'</code> , <code>'ou=users,dc=example,dc=com'</code>
<code>ldap.search.filter</code>	<code>'(uid={0})'</code>	The filter expression used in the user search
<code>ldap.search.derefLink</code>	<code>false</code>	Enables/disables link dereferencing during the search
<code>ldap.search.timeLimit</code>	<code>0 (unlimited)</code>	The time to wait before the search fails
<code>ldap.search.attributesToReturn</code>	<code>null (all)</code>	The attributes to return as part of the search
<code>ldap.authenticator.useBind</code>	<code>true</code>	if <code>true</code> uses a <code>BindAuthenticator</code> to bind as the authenticating user, if <code>false</code> uses a <code>PasswordComparisonAuthenticator</code> to lookup the user login name and compare passwords
<code>ldap.authenticator.attributesToReturn</code>	<code>null (all)</code>	names of attribute ids to return; use <code>null</code> to return all and an empty list to return none
<code>ldap.authenticator.dnPatterns</code>	<code>null (none)</code>	optional pattern(s) used to create DN search patterns, e.g. <code>["cn={0},ou=people"]</code>

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 requests
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 PartialResultExceptions 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. Changing this is NOT recommended as it can have unforeseen consequences within the security filters.
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
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

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