LDAP authentication and authorisation, part 2: Configuring Search Guard

jkressin · 2019-07-03

In the previous article, we have installed and configured OpenLDAP and created some users and groups. In this article we show how to configure Search Guard to use LDAP for authentication and authorisation.

LDAP Authentication

We start with a fresh Quickstart Installation of Elasticsearch and Search Guard. The demo configuration already contains an LDAP section which is disabled by default. In order to use the OpenLDAP server for authentication we change the configuration like:

```
authc:
  ...
  ldap:
    http_enabled: true
    order: 1
    ...
    authentication_backend:
      config:
        ...
        hosts:
          - ldap.example.com:389
        bind_dn: cn=admin,dc=example,dc=com
        password: pass
        userbase: 'dc=example,dc=com'
        usersearch: '(uid={0})'
        username_attribute: uid
```

hosts

Defines the LDAP host and port Search Guard should connect to. You can define more than one LDAP servers in this section. This is used for fail-over: If one LDAP server is not reachable, Search Guard will try

to connect to the next server in the list.

bind_dn and password

This defines the distinguished name and the password of the account that Search Guard uses to connect to the LDAP server. This account does not need to have any admin privileges, it just needs to have permission to perform LDAP searches. If your LDAP server supports anonymous access, you can also skip these settings.

userbase

This defines the subtree of the LDAP directory where the users are stored. In the OpenLDAP installation we have set up in the previous article all users are stored in the dc=example,dc=com subtree, so we configure this DN here. During authentication, Search Guard will query all entries in the configured userbase and its subtrees.

usersearch

This is the actual LDAP query that Search Guard issues against the userbase to look up the user that wants to authenticate. The variable {0} is replaced with the username, so if the username is 'hr*employee', Search Guard will look for any entry in and underneath the dc=example,dc=com LDAP subtree where the uid attribute equals hr*employee. For a standard Active Directory configuration, the default usersearch is (sAMAccountName={0}).

username_attribute

This defines the attribute of the LDAP user entry that Search Guard will retrieve the username from. In most cases, this will be the attribute we used for the usersearch. However, you can choose any attribute here as username. This username can then be leveraged for mapping the user to Search Guard roles.

After making these changes, do not forget to reload cluster configuration with:

```
./sgadmin.sh \
  -cd ../sgconfig/ -icl -nhnv \
  -cacert ../../../config/root-ca.pem \
  -cert ../../../config/kirk.pem \
  -key ../../../config/kirk-key.pem
```

Let us now test the setup with the authinfo endpoint:

```
curl -k 'https://localhost:9200/_searchguard/authinfo?pretty' -u hr_employee:pass
```

Result:

```
{
  "user" : "User [name=hr_employee, roles=[], requestedTenant=null]",
  "user_name" : "hr_employee",
  "user_requested_tenant" : null,
  "remote_address" : "[::1]:53991",
  "backend_roles" : [ ],
  "custom_attribute_names" : [
    "ldap.dn",
    "attr.ldap.hasSubordinates",
    "attr.ldap.cn",
    "attr.ldap.entryUUID",
    ...
}
```

As you can see, the authentication of the user hr_employee succeeded. The only thing lacking are the user's backend roles. Our goal is to use LDAP groups for the role mapping. So the next step is to setup authorization in order to fetch the users groups from OpenLDAP.

LDAP Authorization

There are two possible setups to store group membership information within an LDAP server. Group membership can be either stored as an attribute of a user, or a group entry can store information about its members. In our demo, we have chosen the latter case. Please note that Search Guard supports both cases and both are described in the documentation.

We will configure the Search Guard plugin so that it runs an ldapsearch similar to:

```
ldapsearch -h ldap.example.com \
  -p 389 \
  -D "cn=admin,dc=example,dc=com" \
  -W \
  -b "dc=example,dc=com" \
  '(member=uid=hr_employee)'
```

This will search for all group entries with a member attribute equal to uid=hr*employee. In other words, it will search for all LDAP groups for the user hr*employee.

Below we present LDAP authorization parameters that need to be altered in the Quickstart installation:

```
authz:
  roles_from_myldap:
    http_enabled: true
    ...
    authorization_backend:
      config:
        ...
        hosts:
          - ldap.example.com:389
        bind_dn:  cn=admin,dc=example,dc=com
        password: pass
        rolebase: 'dc=example,dc=com'
        rolesearch: '(member=uid={1})'
        rolename: cn
        userbase: 'ou=people,dc=example,dc=com'
        usersearch: '(uid={0})'
```

Most entries are similar to ones we have configured for authentication. The new ones are:

- roleasearch - the LDAP query to fetch roles,
- rolebase - subtree of directory service to search for roles.
- rolename - the attribute of the group entry from which the role name is fetched. If not configured, the complete DN of the group entry is used.

As with the usersearch, the rolesearch can contain variables which get substituted before issuing the LDAP query for groups. The variable {1} is substituted with the username from the authentication step. So with the configuration above, Search Guard will look up all groups in the ou=people,dc=example,dc=com subtree where the member attribute has the value uid=hr_employee.

With our OpenLDAP installation, this will return the group entry with the DN cn=SearchGuard,dc=example,dc=com:

Since we configured the rolename to be fetched from the cn attribute, our user will have the backend role SearchGuard. We can now use this backend role to assign one or more Search Guard roles in sg*roles*mapping.yml:

```
my_sg_role:
  backendroles:
    - SearchGuard
```

Again, the Search Guard configuration needs to be reloaded with sgadmin command.

Then we request authinfo endpoint with:

```
curl -k 'https://localhost:9200/_searchguard/authinfo?pretty' -u hr_employee:pass
{
  "user" : "User [name=hr_employee, roles=[SearchGuard], requestedTenant=null]",
  "user_name" : "hr_employee",
  "user_requested_tenant" : null,
  "remote_address" : "[::1]:63139",
  "backend_roles" : [
    "SearchGuard"
  ],
  "sg_roles" : [
    "my_sg_role"
  ],
  ...
}
```

This time the hr*employee user also has the SearchGuard backend role which was fetched from the LDAP group. In addition, the user has the my*sg_role Search Guard role. This means the user has been both authenticated and authorized using LDAP, and that the role mapping works as expected.

Where to go next

- Read the previous article in this series about how to configure OpenLDAP
- Read the LDAP authentication configuration guidelines in our docs
- Read the LDAP authorisation guidelines in our docs
- Download the Search Guard Active Directory / LDAP presentation

Image: shutterstock / Profit_Image

This article was published on the Search Guard blog: https://search-guard.com/blog/elasticsearch-ldap-authentication-authorisation/