Security

Stardog's security model in detail

Taught by:

Stephen Nowell
Director, Technical Support

Learning Objectives

- Stardog's Role-based Access Control implementation
- How to create new Users and Roles, assigning permissions to each
- The specifics and unique qualities of Named Graph Security
- The basics of LDAP integration
- How to understand the requirements for running Stardog with SSL

Getting Started

Introduction

- Stardog implements a security protocol based on Apache Shiro
- Authentication
  - User/Password
  - LDAP integration
  - Kerberos integration
- Authorization
  - Role-Based Access Control (RBAC) model
  - Access Control List (ACL) for databases, database metadata, named graphs, and admin commands

Authentication

- Default Authentication
  - Stardog manages user accounts and passwords internally
  - Users and Roles granted access to Resources
  - Users can be enabled / disabled
  - Password requirements set at server level
    - password.length.min
    - password.length.max
    - password.regex
- LDAP/Kerberos Authentication
  - Stardog delegates authentication of user accounts to LDAP/Kerberos system
  - Stardog continues to manage User/Role access to Resources

LDAP

```
dn: cn=stardogSuperUsers,ou=group,dc=example,dc=com
cn: stardogSuperUsers
objectclass: groupOfNames
member: uid=superuser,ou=people,dc=example,dc=com

dn: cn=stardogModelers,ou=group,dc=example,dc=com
cn: stardogModelers
objectclass: groupOfNames
member: uid=hank,ou=people,dc=example,dc=com

dn: cn=stardogReaders,ou=group,dc=example,dc=com
cn: stardogReaders
objectclass: groupOfNames
member: uid=beth,ou=contractors,dc=example,dc=com
```

Credentials and other user information are stored as usual:

```
dn: uid=superuser,ou=people,dc=example,dc=com
objectClass: inetOrgPerson
cn: superuser
sn: superuser
uid: superuser
userPassword: superpassword

dn: uid=hank,ou=people,dc=example,dc=com
objectClass: inetOrgPerson
cn: hank
sn: hank
uid: hank
userPassword: hankpassword

dn: uid=beth,ou=contractors,dc=example,dc=com
objectClass: inetOrgPerson
cn: beth
sn: beth
uid: beth
userPassword: bethpassword
```

```
security.realms = ldap
ldap.provider.url = ldap://localhost:5860
ldap.security.principal = uid=admin,ou=people,dc=example,dc=com
ldap.security.credentials = secret
ldap.user.dn.templates = "uid={0},ou=people,dc=example,dc=com","uid={0},ou=contractors,dc=example,dc=com"
ldap.superusers.group = cn=stardogSuperUsers,ou=group,dc=example,dc=com
ldap.role.mappings = "modelers":"cn=stardogModelers,ou=group,dc=example,dc=com","queryers":["cn=stardogReaders,ou=gr
ldap.member.attributes = member
ldap.cache.invalidate.time = 1h
```

Kerberos

Property	Description
`krb5.keytab`	The path to the keytab file for the Stardog server.
`krb5.admin.principal`	The Kerberos principal that will be the default administrator of is service.
`krb5.debug`	A boolean value to enable debug logging in the Java Kerberos libraries.
`krb5.user.translation.regex`	A string value used to translate a krb5 principal name to a Stardog username. The string is an expression in two parts divided by a `:`. On the left side is a matching regex of the krb5 principal name to replace and on the right side is the string to replace it with. By default this is `/:-`. This means "replace any `/` character in the krb5 principal with a `-` character and use that as the Stardog username". Thus the krb5 principal name `stardog/admin` will be translated to `stardog-admin`. The details of the substitution rules are that of Java `String.replaceAll()`.
`pack.krb5.principal`	The Kerberos principal that is authorized to connect as a cluster peer. Stardog cluster nodes connected directly to each other. This directive tells Stardog to use Kerberos authentication for this communication and to only allow connections from entities with the given Kerberos principal
`pack.krb5.keytab`	The path to the keytab file that Stardog cluster peers will use to prove to other nodes they are authorized peers. The principal in this keytab must match the value of `pack.krb5.principal`.

Authorization

- Users and Roles are granted specific access to Resources
- While explicit grants to individual Users are allowed, it is highly encouraged to utilize Roles as much as possible
  - Easily grant batches of permissions to a given User by assigning a Role
  - Easily know what permissions Users have by looking at Roles

Actions and Resources

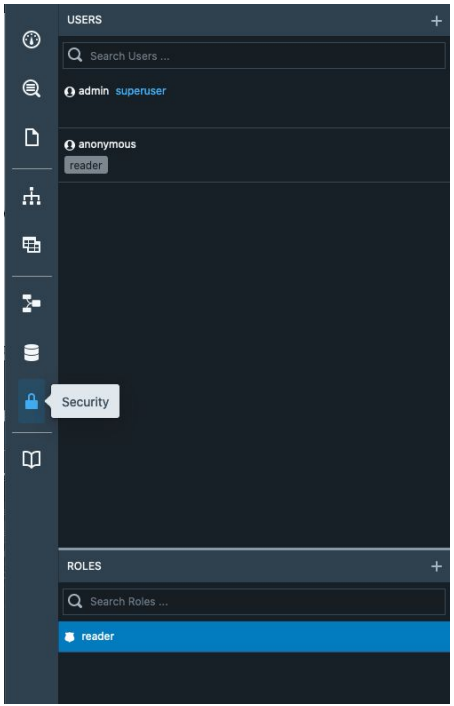
Actions	Resources
• CREATE • READ • WRITE • DELETE • GRANT • REVOKE • EXECUTE • ALL	• DB • User • Role • Admin • Metadata • Named Graph • Virtual Graph • Data Source • ICV Constraints

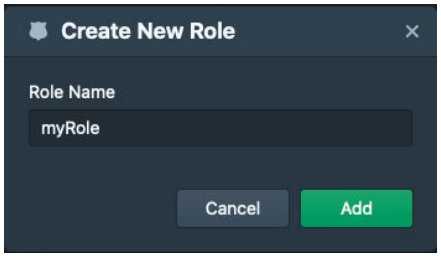
Any Action can be granted over any Resource

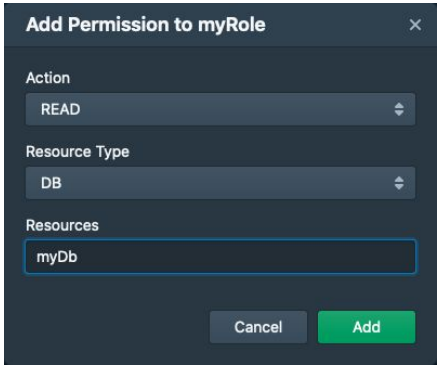
Some combinations, while allowed, are meaningless

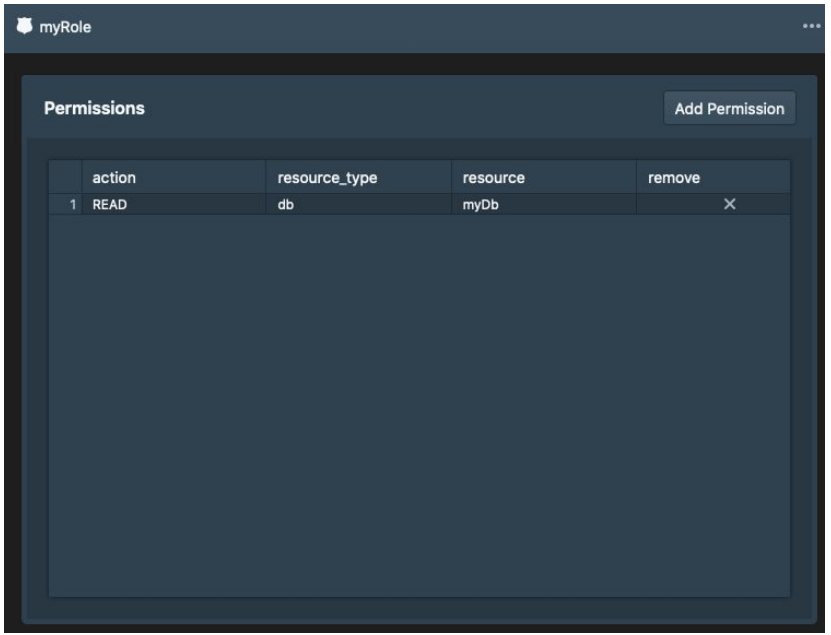
Creating Users & Roles Demo

Creating Users and Roles









Permissions

Anatomy of a Permission

- Permissions consist of a Subject, Action, and Resource
  - ***Subject*** can perform ***Action*** over ***Resource***
- Subject is either a User or a Role
- Resource consists of both a type and a name or wildcard
  - [CREATE, “db:*”]
  - [READ, “db:myDb”]
  - [GRANT, “virtual-graph:myVG”]
  - [READ, “named-graph:myDb\http://stardog.com/graphs/graph1”]

Permissions

	DB	User	Role	Admin	Metadata	Named Graph	Virtual Graph	Data Source	ICV Constraints
CREATE	✅	✅	✅	❌	❌	❌	✅	✅	❌
READ	✅	✅	✅	❌	✅	✅	✅	✅	✅
WRITE	✅	✅	❌	❌	✅	✅	☑️	✅	✅
DELETE	✅	✅	✅	❌	❌	❌	✅	✅	❌
GRANT	✅	❌	❌	✅	✅	✅	✅	✅	✅
REVOKE	✅	❌	❌	✅	✅	✅	✅	✅	✅
EXECUTE	❌	✅	❌	✅	❌	❌	❌	❌	❌

✅ - Valid Permission ☑️ - Potentially valid in future ❌ - Invalid

Permissions

	DB	User	Role	Admin	Metadata	Named Graph	Virtual Graph	Data Source	ICV Constraints
CREATE	✅	✅	✅	❌	❌	❌	✅	✅	❌

- [CREATE, “type:*”] - Create instances of a given Resource type
- [CREATE, “virtual-graph:*”]
  - With [READ, “data-source:mySource”] - Add a VG with given data source
  - With [CREATE, “data-source*”] - Add a VG with a private data source
  - With [DELETE, “virtual-graph:myVg”] - Update an existing VG

✅ - Valid Permission ☑️ - Potentially valid in future ❌ - Invalid

Permissions

	DB	User	Role	Admin	Metadata	Named Graph	Virtual Graph	Data Source	ICV Constraints
READ	✅	✅	✅	❌	✅	✅	✅	✅	✅

- [READ, “db:myDb”] - Database appears in list of databases. Full query access (except wrt NG security). Access to BITES document store count and documents
- [READ, "user:myUser”] - User appears in list of users. Can see enabled status, permissions, and roles
- [READ, "role:myRole”] - Role appears in list of roles. Can see effective permissions
- [READ, "metadata:myDb”] - Read access to database properties/settings
- [READ, "named-graph:myDb\myGraph”] - When NG security enabled, read access to triples in the graph
- [READ, "virtual-graph:myVG”] - VG appears in list of VGs. Full read access. Get options/mappings/ Generate data model
- [READ, "data-source:mySource”] - Data source appears in list of data sources. Can create VG backed by it
- [READ, "icv-constraints:myDb”] - List and show a database’s ICV constraints

✅ - Valid Permission ☑️ - Potentially valid in future ❌ - Invalid

Permissions

	DB	User	Role	Admin	Metadata	Named Graph	Virtual Graph	Data Source	ICV Constraints
WRITE	✅	✅	❌	❌	✅	✅	☑️	✅	✅

- [WRITE, "db:myDb”] - Perform SPARQL Update queries and SNARL (Java) update operations. Add/remove/clear documents from BITES store
- [WRITE, "user:myUser”] - Change user’s password
- [WRITE, "metadata:myDb”] - Modify database properties/settings. Add/remove stored queries
- [WRITE, "named-graph:myDb\myGraph”] - When NG security enabled, write access to the graph
- [WRITE, "icv-constraints:myDb”] - Modify and delete a database’s ICV Constraints

✅ - Valid Permission ☑️ - Potentially valid in future ❌ - Invalid

Permissions

	DB	User	Role	Admin	Metadata	Named Graph	Virtual Graph	Data Source	ICV Constraints
DELETE	✅	✅	✅	❌	❌	❌	✅	✅	❌

- [DELETE, “type:*”] - Delete instances of a given Resource type
  - Can also explicitly name DBs, Users, etc.
  - With [CREATE, "virtual-graph:*”] - Update a given VG
  - With [CREATE, "data-source:*”] - Update a given Data source

✅ - Valid Permission ☑️ - Potentially valid in future ❌ - Invalid

Permissions

	DB	User	Role	Admin	Metadata	Named Graph	Virtual Graph	Data Source	ICV Constraints
GRANT	✅	❌	❌	✅	✅	✅	✅	✅	✅
REVOKE	✅	❌	❌	✅	✅	✅	✅	✅	✅

- [GRANT, “type:name*”] - Add other users to the ACL of a given resource
  - IFF the user doing the granting has the permission being granted
- [REVOKE, “type:name”] - Remove other users from the ACL of a given resource

✅ - Valid Permission ☑️ - Potentially valid in future ❌ - Invalid

Permissions

	DB	User	Role	Admin	Metadata	Named Graph	Virtual Graph	Data Source	ICV Constraints
EXECUTE	❌	✅	❌	✅	❌	❌	❌	❌	❌

- [EXECUTE, "user:myUser”] - Special case allowing a user to run a command impersonating another user
  - Allows an admin user to test read/write access of a given user

✅ - Valid Permission ☑️ - Potentially valid in future ❌ - Invalid

DBMS Admin Permissions

Certain actions require special permissions that can only be set outside of Stardog Studio

- [EXECUTE, "dbms-admin:myDb”] - Verify, Optimize, Offline/Online a database
- [EXECUTE, "dbms-admin:shutdown”] - Explicit permission to remote shutdown a server
- [READ, "dbms-admin:set-property”] - Read server/system properties
- [WRITE, "dbms-admin:set-property”] - Update server/system properties

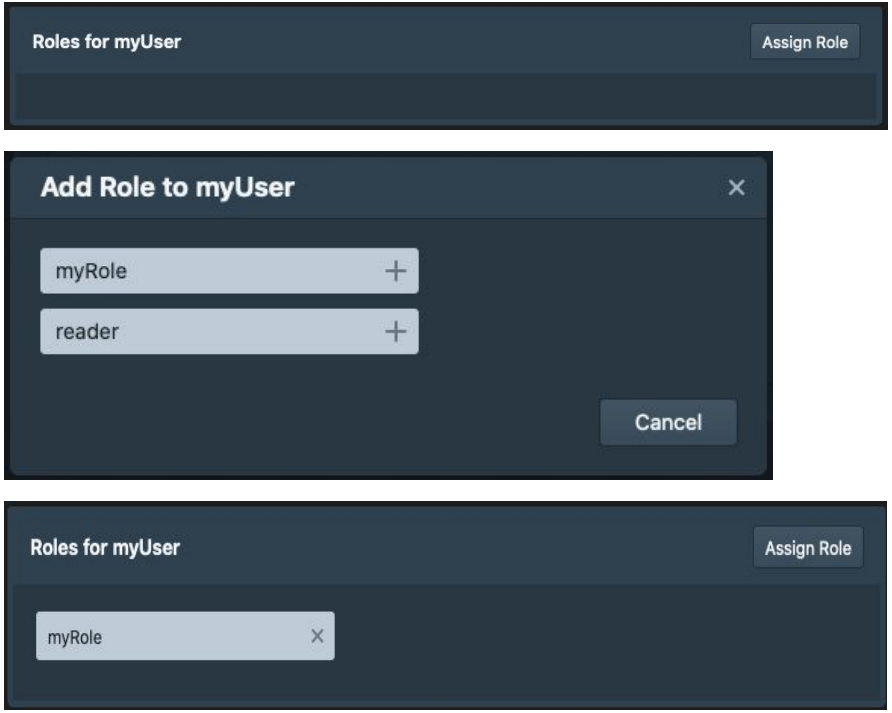
Superuser

Certain actions can only be performed by a Superuser

- Any sort of cache management for a cluster
- Enable/Disable a User
- View a Role's permissions if not assigned that Role
- View Users assigned to a given Role
  - Note that viewing Roles assigned to a given User only requires READ on the User
- Convert superusers into regular Users
- Open a connection on another User's transaction

Assigning Roles to Users Demo

Assigning Roles to Users



Named Graph Security

Named Graph Security

- Must explicitly be enabled via the `security.named.graphs` configuration option
- Users receive query results based on the set of all named graphs in a given database that they have READ access to
- By default, if a user does not have READ/WRITE access to any named graphs in a given database, it will appear empty and unwriteable
    - Server can be configured to interpret an empty list as allowing access to ALL graphs (i.e., no restrictions have been specified)
- **Important:** When named graph security is enabled, a user will need the additional permission of [READ, "named-graph:virtual://myVg”] in order to query it, because of the SPARQL syntax needed

Named Graph Security Demo

NGS LIVE DEMO

SSL

SSL

- By default a Stardog server will support http only
- In most production environments SSL will be required
- Requirements
  - A .jks keystore containing a valid SSL certificate for the server
  - A .jks trust store containing the same certificate
- On the Stardog server
  - Add java SSL properties to $STARDOG_HOME/stardog.properties
    - javax.net.ssl.keyStore=/path/to/my-keystore.jks
    - javax.net.ssl.keyStorePassword=my-very-secure-password
  - When running `server start` command (either manually or via systemd), pass the --enable-ssl flag to support both http and https or --require-ssl for https only
    - --port and --ssl-port can also be set in this same command
    - Default ports are http on 5820 and https on 5821

SSL

- On the Stardog client
  - In a CLI environment
    - Point the environment to the truststore via an environment variable
    - `export STARDOG_JAVA_ARGS="-Djavax.net.ssl.trustStore=/path/to/my-truststore.jks -Djavax.net.ssl.trustStorePassword=my-very-secure-password
    - Will likely also want to set `-Dstardog.default.cli.server` in the java args to point to the https endpoint

Learning Objectives

Learning Objectives

- Stardog's Role-Based Access Control implementation
- How to create new Users and Roles, assigning permissions to each
- The specifics and unique qualities of Named Graph Security
- The basics of LDAP integration
- How to understand the requirements for running Stardog with SSL

Thank you