Security

What are we going to do?

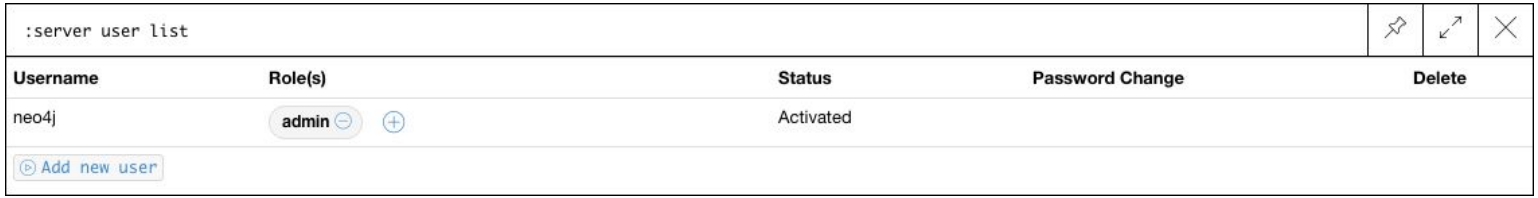
- Role based access control
- Procedure annotated access control

Role Based Access Control

Create a user for yourself

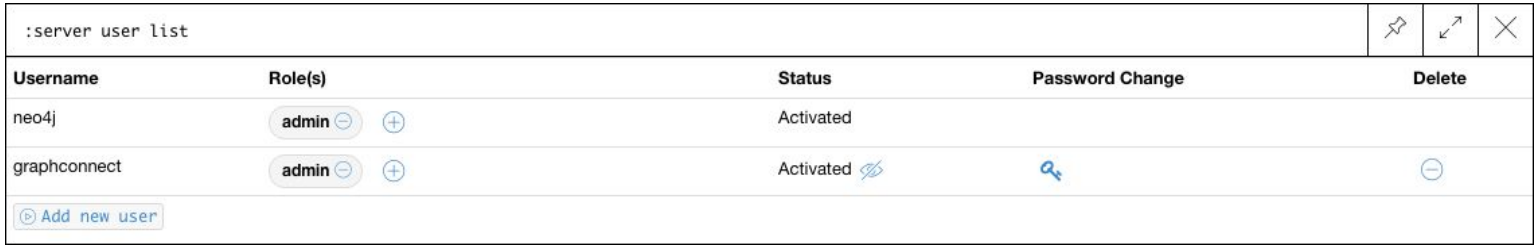
Type the following command in the query pane:

```
:server user list
```



Create a user for yourself

Now add yourself as a new user and assign yourself the admin role.



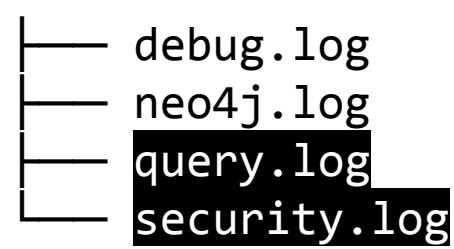
Security logging

- Enable query logging and then restart Neo4j `dbms.logs.query.enabled=true`
- Open two browser windows
  - Note: credentials are cached in the browser, so it's easiest to use Incognito windows so that they are independent from each other
- Log into different users in each window
- Run queries
- Inspect the security log for users and their respective queries

Security logging


```
logs/
├── debug.log
├── neo4j.log
├── query.log
└── security.log
```

Security logging



logs/security.log

```
2016-10-05 16:16:22.045+0000 INFO  [AsyncLog @ 2016-10-05 16:16:22.045+0000]  [neo4j]: logged in
2016-10-05 16:16:22.045+0000 INFO  [AsyncLog @ 2016-10-05 16:16:22.045+0000]  [neo4j]: logged in
2016-10-05 16:16:22.088+0000 INFO  [AsyncLog @ 2016-10-05 16:16:22.088+0000]  [neo4j]: logged in
2016-10-05 16:16:26.256+0000 INFO  [AsyncLog @ 2016-10-05 16:16:26.256+0000]  [neo4j]: logged in
2016-10-05 16:16:26.260+0000 INFO  [AsyncLog @ 2016-10-05 16:16:26.260+0000]  [neo4j]: logged in
2016-10-05 16:16:50.354+0000 INFO  [AsyncLog @ 2016-10-05 16:16:50.354+0000]  [neo4j]: logged in
2016-10-05 16:16:50.357+0000 INFO  [AsyncLog @ 2016-10-05 16:16:50.357+0000]  [neo4j]: logged in

2016-10-05 16:17:14.021+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.020+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.033+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.033+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.077+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.077+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.078+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.078+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.091+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.091+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.091+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.091+0000]  [graphconnect]: logged in
```

logs/query.log

```
2016-10-05 16:30:19.378+0000 INFO  10 ms: server-session    http   127.0.0.1    /db/data/transaction/commit
graphconnect - EXPLAIN CREATE (:Event {name: "Graph Connect San Francisco 2016"}) - {} - {}

2016-10-05 16:30:19.792+0000 INFO  4 ms: server-session    http   127.0.0.1    /db/data/transaction/33/commit
graphconnect - CREATE (:Event {name: "Graph Connect San Francisco 2016"}) - {} - {}
```

User Roles

Predefined roles

Action	reader	editor	publisher	architect	admin	(no role)
Change own password	X	X	X	X	X	X
View own details	X	X	X	X	X	X
Read data	X	X	X	X	X	
View own queries	X	X	X	X	X	
Terminate own queries	X	X	X	X	X	
Write/update/delete data		X	X	X	X	
Create new types of properties key			X	X	X	
Create new types of nodes labels			X	X	X	
Create new types of relationship types			X	X	X	
Create/drop index/constraint				X	X	
Create/delete user					X	
Change another user's password					X	
Assign/remove role to/from user					X	
Suspend/activate user					X	
View all users/roles					X	
View all roles for a user					X	
View all users for a role					X	
View all queries					X	
Terminate all queries					X	

Predefined roles

Privileges	Reader	Publisher	Architect	Admin
Change own password	•	•	•	•
Read data	•	•	•	•
View own details	•	•	•	•
Terminate own query	•	•	•	•
Write/update/delete data		•	•	•
Manage index/constraints			•	•
View all users and roles				•
Create/delete users				•
Assign/delete user roles				•
Suspend/activate users				•
Terminate others' queries				•
Change others' passwords				•

User roles

```
CALL dbms.security.listRoles()
```

role	users
reader	[]
architect	[]
admin	[graphconnect, neo4j]
publisher	[]

Can read

Can read, write, and execute schema operations

Can do anything

Can read and write

Create a new user with read only permission

Execute the following procedure calls to create a new user and assign them to the ‘reader’ role:

```
CALL dbms.security.createUser("restrictedUser", "neo4j", false)

CALL dbms.security.addRoleToUser("reader", "restrictedUser")
```

User roles

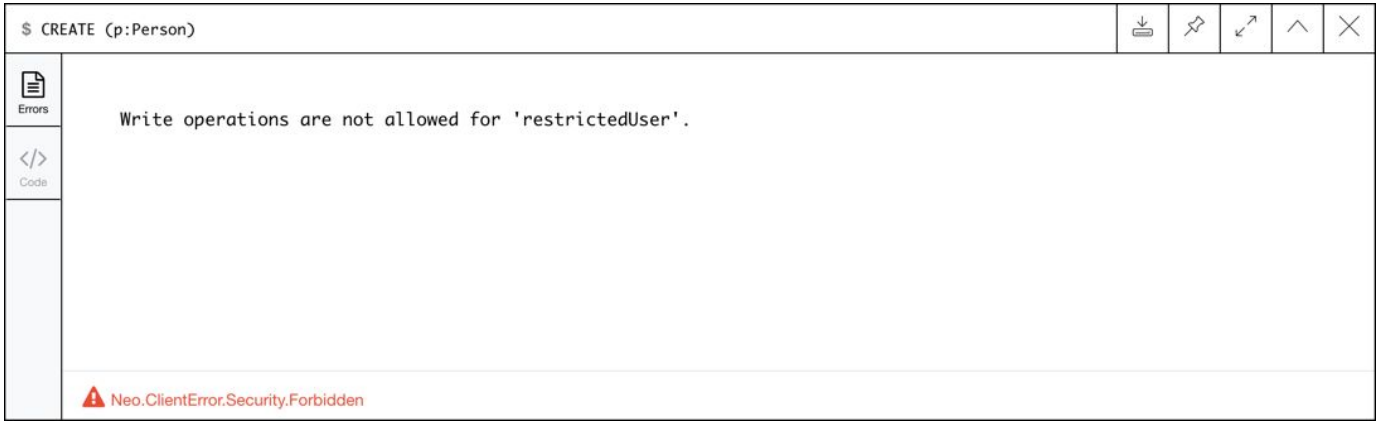
```
CALL dbms.security.listRoles()
```

role	users
reader	[restrictedUser]
architect	[]
admin	[graphconnect, neo4j]
publisher	[]

Create data: denied!

Now login as that user in another browser session and try and create some data:

```
CREATE (p:Person);
```



Procedure Annotated Access Control

Locked down roles

We can also create completely locked down roles which can only make calls to specific procedures.

e.g. imagine we want to create a role which can only find the movies that people have acted in but can’t do anything else!

Create locked down role

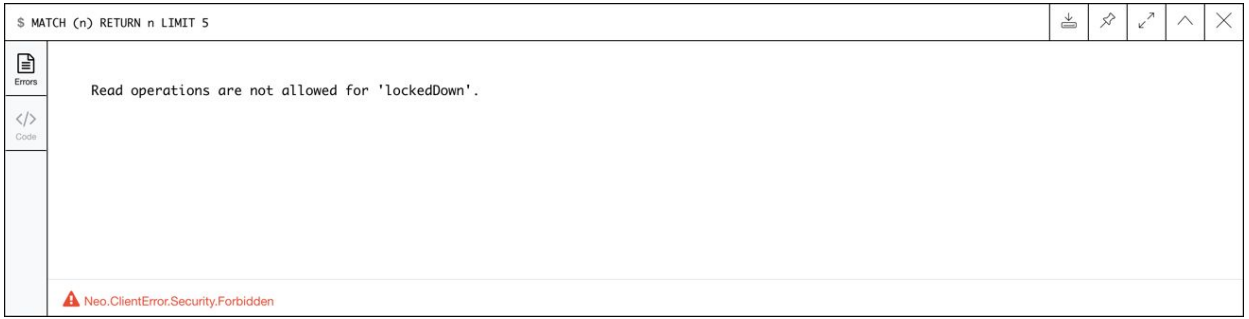
Let’s create a locked down role and a new user to assign to that role. Run the following commands:

```
CALL dbms.security.createRole("movies_only")
CALL dbms.security.createUser("lockedDown", "abc", false)
CALL dbms.security.addRoleToUser("movies_only", "lockedDown")
```

Create locked down role

Login as our new user and try to execute any Cypher query e.g.

```
MATCH (n)
RETURN n
LIMIT 5
```



Our user can’t do anything!

Locked down roles

```
@Context
public GraphDatabaseService db;

@Description( "Find movies by an actor" )
@Procedure( name = "training.moviesOnly", mode = Mode.READ )
public Stream<Movie> moviesOnly( @Name( "name" ) String name )
       throws InvalidArgumentsException, IOException
{
   String query = "MATCH (:Person {name: {name}})-[:ACTED_IN]->(movie) RETURN movie";
   return db.execute( query, map("name", name) )
            .stream()
            .map( Movie::new );
}
```

Locked down roles

```
dbms.security.procedures.roles=
     training.moviesOnly:movies_only;
     training.writeProcedure:allowed_role;
     training.waitFor:allowed_role;
```

Locked down roles

```
dbms.security.procedures.roles=
     training.moviesOnly:movies_only;
     training.writeProcedure:allowed_role;
     training.waitFor:allowed_role;
```

Users with the 'movies_only' role can only execute the 'training.moviesOnly' procedure

Locked down roles

```
dbms.security.procedures.roles=
     training.moviesOnly:movies_only;
     training.writeProcedure:allowed_role;
     training.waitFor:allowed_role;
```

Users with the 'allowed_role' role can only execute the following procedures:

- training.writeProcedure
- training.waitFor

Locked down roles

```
dbms.security.procedures.roles=training.moviesOnly:movies_only;
training.writeProcedure:allowed_role;training.waitFor:allowed_role;
```

Add this to neo4j.conf and restart neo4j

Locked down roles

Try calling our locked down procedure:

```
CALL training.moviesOnly("Halle Berry")
```

Allow by Default

Allow a role to execute a procedure that is not matched by the `dbms.security.procedures.roles` setting.

```
dbms.security.procedures.allowed_default=training.writeProcedure
```

Whitelisting

Whitelisting allows loading of a selection of procedures from a larger library.

```
dbms.security.procedures.whitelist=training.*
```

End of Module
Security (single server)

Questions ?