Chapter 20. August 2018

Welcome to the August 2018 edition of DataStax Developer’s Notebook (DDN). This month we answer the following question(s);

> My company is investigating using DataStax for our new Customer/360 system in our customer call center. I’m a developer, and do not know how to administer DataStax Enterprise, but, I need to know how to set up user authentication and authorization for my programming and unit tests.Can you help ?
>
> *Excellent question ! Setting up authentication and authorization using DataStax Enterprise (DSE) is super easy. Below we detail all of the relevant topics and steps to achieve same, including source code for all. We detail table level access control, and in the event you need it, row level access control.*

Software versions

The primary DataStax software component used in this edition of DDN is DataStax Enterprise (DSE), currently release 6.0.2. All of the steps outlined below can be run on one laptop with 16 GB of RAM, or if you prefer, run these steps on Amazon Web Services (AWS), Microsoft Azure, or similar, to allow yourself a bit more resource.

For isolation and (simplicity), we develop and test all systems inside virtual machines using a hypervisor (Oracle Virtual Box, VMWare Fusion version 8.5, or similar). The guest operating system we use is CentOS version 7.0, 64 bit.

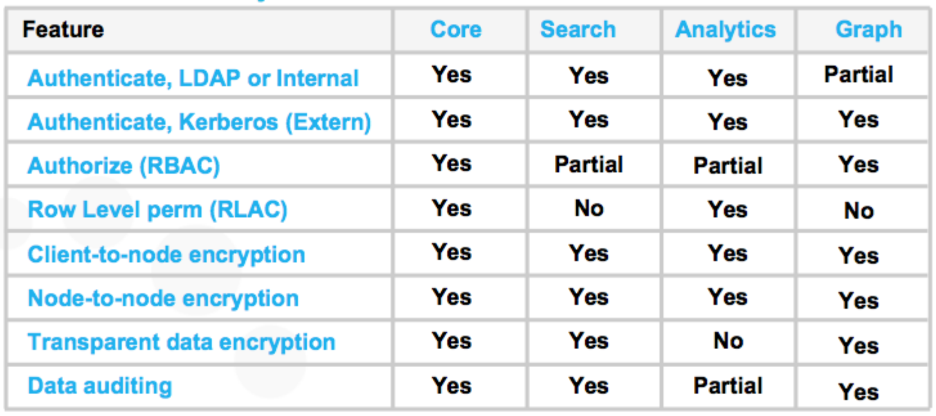
20.1 Terms and core concepts

As stated above, ultimately the end goal is to implement end user authentication and authorization using DataStax Enterprise (DSE). That choice leaves out topics like implementing inter-nodal encryption, other.

As such, we state:

- Securing DSE involves settings at multiple levels; pre-DSE (at the operating system level), and within DSE.
- Within just DSE, there is:
  - Securing database connections (end user connections)
  - (Encrypting) data in flight; between the end user and DSE, between DSE nodes, other
  - (Encrypting) data at rest; on the hard disk
  - Security configuration and reporting files, temporary files, other
  - And other topics

Figure 20-1details an overview of security using DSE. A code review follows.



Feature	Core	Search	Analytics	Graph
Authenticate, LDAP or Internal	Yes	Yes	Yes	Partial
Authenticate, Kerberos (Extern)	Yes	Yes	Yes	Yes
Authorize (RBAC)	Yes	Partial	Partial	Yes
Row Level perm (RLAC)	Yes	No	Yes	No
Client-to-node encryption	Yes	Yes	Yes	Yes
Node-to-node encryption	Yes	Yes	Yes	Yes
Transparent data encryption	Yes	Yes	No	Yes
Data auditing	Yes	Yes	Partial	Yes

Figure 20-1 Security using DSE 6.0.

Relative to Figure 20-1, the following is offered:

- Essentially, row based access control is not available in release 6.0 of DSE when using DSE Search or DSE Analytics queries. (Table based yes, row based no.)

– Further, the audit secure subsystem does not currently have the ability to report an each transform or related, that might be requested inside a DSE Analytics Job.

– Table level access control is available, but not always row based (subset of table) access control.

DSE Authentication

At a high level, authentication is that means by which a user is proven to be who they say they are. Contrast this with, authorization, which is; now that we know who you are, what are you allowed to do.

DataStax Enterprise (DSE) offers DSE unified authentication for database connections only. Comments include:

– Inter-nodal communication (gossip, other) is authentication using SSL certificates

– Database connections (client to node, tools and applications), use DSE unified authentication (DUA). DUA can be implemented using:

  • Internal authentication, is provided directly by DSE using internally maintained passwords, user names, other

  • DSE also support authentication using LDAP, including Active Directory

  • And DSE supports authentication using Kerberos; PKI, KMIP

  • And all of the above schemes (internal, LDAP, Kerberos), are abstracted so that using one scheme, is sufficient for development, and testing, other.

Reference Urls include:

– DSE inter-nodal traffic authentication using SSL is detailed at,

  `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secInternodeSsl.html`

– Setting up Kerberos for use with DSE is detailed at,

  `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secKerberosTOC.html`

– Setting up LDAP for use with DSE is detailed at,

  `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secLDAPScheme.html`

Per the document listed above, DSE version 6.0 is compatible with: Microsoft Active Directory (Windows 2008, Windows 2012), OpenLDAP 2.4.x, and Oracle Directory Server Enterprise Edition 11.1.1.7.0

- Securing (again, SSL), JMX communications is detailed at,

  `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secureJconsoleSSL.html`

- And securing (standard client to server) communications using SSL is detailed at,

  `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/encryptClientNodeSSL.html`

DSE Authorization

In the section above, authentication, the user identity has been confirmed; now, what should this same user be able to perform-

Comments:

- Role based access control (RBAC), is enabled using DSE only if authentication is enabled.
- Using DSE internal authentication, there is a one to one mapping of user names to roles. In short this becomes;
  - Create a role, for example: operator, senior operator, other
  - Create users and passwords
  - Make users members of one or more defined roles
- Using LDAP, there is a one to many mapping; users are assigned roles that match groups inside the LDAP (directory)
- Specific to DSE Search-
  - DSE Search indexes can offer specific/additional authorizations detailed at,

    `http://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secSearchIndexPermissions.html`

    This topic is not detailed further in this document.
  - Setting row level permissions with row level access control (RLAC) is not supported for use with DSE Search or DSE Graph.
- Specific to DSE Graph-
  - Additional restriction to DSE Graph (data) is achieved by setting permissions at the (DSE Graph) keyspace level, detailed here,

  `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secRbacGraph.html`

- Setting row level permissions with row level access control (RLAC) is not supported for use with DSE Search or DSE Graph.

Other DSE Security Topics-

DataStax Enterprise offers transparent data encryption (TDE) on the following:

- Entire tables, except for partition keys which are always stored in plain text)
- SSTables containing data, including system tables, such as system.batchlog and system.paxos)
- Search indexes
- File-based hinted handoff data
- Commit logs
- Sensitive properties in dse.yaml and cassandra.yaml
- TDE only applies to data stored in the database. DSE does not support encrypting data that is used by Spark and stored in DSEFS or local temporary directories.
- For DSE Graph, cached data is not encrypted. Encryption may slightly impact performance
- (Encryption) is detailed further at,
  - `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secEncryptTDE.html`
  - `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secEncryptSearch.html`

As a topic, securing DSE ports is detailed at,

- `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secFirewallPorts.html`
- `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secTmp.html`

DSE offers an audit secure subsystem. Comments include:

- This subsystem is very much any similarly named subsystem found in relational databases, other. This subsystem records queries and prepared statements from DataStax drivers to a (log file).
- Details are located at,

https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secAuditTOC.html

https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secAuditLogFormat.html

https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secAuditTableColumns.html

DSE Authorization Object Hierarchy

Figure 20-2 details the DSE authorization objects hierarchy. A code review follows.

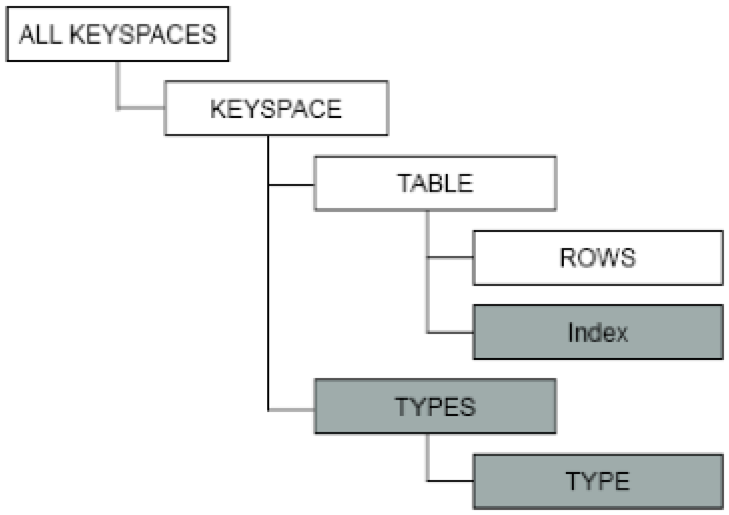


Figure 20-2 DSE object authorization hierarchy

Relative to Figure 20-2, the following is offered:

- In the DataStax Enterprise (DSE) object hierarchy, tables are wholly contained inside one keyspace, and a keyspace serves as the most identifiable storage object, providing; replication factor, node placement, logging constraints, other.
- DSE offers table level access control, and for most of DSE (exceptions noted above), row level access control.
- Further detail is available at the following Url,

  `https://docs.datastax.com/en/dse/6.0/dse-admin/datastax_enterprise/security/secDataResourcesAbout.html`

Similar to most relational databases, DataStax Enterprise (DSE) controls authorization via sets of GRANT and REVOKE statements, on a member of the object hierarchy listed above.

20.2 Complete the following

At this point in this document we have completed a high level overview of DataStax Enterprise authentication and authorization. In this section, we detail the steps to implement same. Comments:

- The following assumes a single node operating DataStax Enterprise (DSE) cluster. Commands are listed for CQLSH, but should work in the client side driver, DSE Studio, other, without grief.
- All of the (changes, implementing security), can be implemented without DSE cluster downtime. For ease and brevity, these instructions call to reboot the single node as assumed. Why? If you do make an error in the cassandra.yaml file, any mistake will be immediately obvious; a nice (debugging) aid.

20.2.1 Create the following CQL artifacts

So that we have something to secure, execute the following displayed in Example 20-1. A code review follows.

Example 20-1 CQL assets to secure

```
DROP KEYSPACE IF EXISTS ks_6221;
   //
CREATE KEYSPACE ks_6221 WITH REPLICATION =
   {'class': 'SimpleStrategy', 'replication_factor': 1};
      //
USE ks_6221;

DROP TABLE IF EXISTS cust_orders;
   //
CREATE TABLE cust_orders
   (
   region            TEXT,
   cust_name         TEXT,
   ord_num           INT,
   other             TEXT,
   PRIMARY KEY ((region, cust_name), ord_num)
   );
INSERT INTO cust_orders (region, cust_name, ord_num, other)
   VALUES ('EMEA', 'IKEA'  , 101, 'Shoes'           );
INSERT INTO cust_orders (region, cust_name, ord_num, other)
```

```
   VALUES ('NA'  , 'SEARS' , 101, 'Shoes, Washer'   );
INSERT INTO cust_orders (region, cust_name, ord_num, other)
   VALUES ('NA'  , 'SEARS' , 102, 'Oranges'          );
INSERT INTO cust_orders (region, cust_name, ord_num, other)
   VALUES ('NA'  , 'MACYS' , 101, 'Dress, Tie'      );

DROP TABLE IF EXISTS cust_payments;
   //
CREATE TABLE cust_payments
   (
   cust_name         TEXT,
   payment_num       INT,
   other             TEXT,
   PRIMARY KEY ((cust_name, payment_num))
   );
INSERT INTO cust_payments (cust_name, payment_num, other)
   VALUES ('SEARS' , 101, '$100'          );
INSERT INTO cust_payments (cust_name, payment_num, other)
         VALUES ('MACYS' , 101, '$200'          );
```

Relative to Example 20-1, the following is offered:

- We make a keyspace and two tables, add data.
- Note the primary keys of both tables.

20.2.2 Edit the cassandra.yaml

Confirm or set the following in the cassandra.yaml file:

```
# default, confirm set
authenticator: com.datastax.bdp.cassandra.auth.DseAuthenticator
#
# default, confirm set
role_manager: com.datastax.bdp.cassandra.auth.DseRoleManager
#
#  Uncomment any that are commented
#  permissions_cache_max_entries  not present in 6.0 DSE
permissions_validity_in_ms: 2000
permissions_uddate_interval_in_ms: 2000
permissions_cache_max_entries: 1000
```

Reference Urls for each above are located at,

`https://docs.datastax.com/en/dse/5.1/dse-admin/datastax_enterprise/security/secRlac.html`

`https://docs.datastax.com/en/dse/5.1/dse-admin/datastax_enterprise/security/secAuthCacheSettings.html#secAuthCacheSettings__cache`

20.2.3 Edit the dse.yaml

Confirm or set the following in the cassandra.yaml file:

```
#  Uncomment all
#  Set,      enabled: true
#  Ensure,   default_scheme: internal
authentication_options:
  enabled: true
  default_scheme: internal
  other_schemes:
  scheme_permissions: false
  allow_digest_with_kerberos: true
  plain_text_without_ssl: warn
  transitional_mode: disabled
#  Uncomment
role_management_options:
  mode: internal
#  Uncomment
#  Set,      enabled: true
authorization_options:
  enabled: true
  transitional_mode: disabled
  allow_row_level_security: true
```

Reference Urls for each above are located at,

`https://docs.datastax.com/en/dse/5.1/dse-admin/datastax_enterprise/config/configDseYaml.html#configDseYaml__allow_row_level_security`

20.2.4 Reboot DSE

Again, not required, and certainly not required for production. The reason we perform this step here, is because the boot screen will throw an error immediately, should an error have been introduced (in the edits to these two files).

If nothing else; consider this a strong debugging step.

20.2.5 Test the Work From Above, and Secure a Little More

After the work above, it should fail when you try to access DSE using CQLSH,

```
cqlsh

>>Connection error: ('Unable to connect to any servers',
{'127.0.0.1': AuthenticationFailed('Remote end requires
authentication',)})
```

`Login to CQLSH and run the code as shown below,`

```
cqlsh -u cassandra -p cassandra

CREATE ROLE dse_admin with SUPERUSER = true AND LOGIN = true

   and PASSWORD = 'password';

exit


# Bad, shows password in ps -ef

cqlsh -u dse_admin -p password

# Preferred, will be prompted for password

cqlsh -u dse_admin

   LIST ROLES;


   role      | super | login | options

   -----------+-------+-------+---------

   cassandra |  True |  True |        {}

   dse_admin |  True |  True |        {}

   (2 rows)
```

```
//  Safe to move forward
DROP ROLE cassandra;
LIST ROLES;
role      | super | login | options
-----------+-------+-------+---------
dse_admin |  True |  True |        {}
(1 rows)
```

In the code fragment above, we performed the following:

- We removed the default/generated cassandra/cassandra user.
- And we added a replacement DSE system administrator with the name and password as shown.‘

20.2.6 Create Security Constraints

Example 20-2 lists the security constraints code we want to put in place. A code review follows.

Example 20-2 Creating security constraints, code

```
DROP ROLE IF EXISTS bob             ;
DROP ROLE IF EXISTS nancy           ;
DROP ROLE IF EXISTS dirk            ;
   //
DROP ROLE IF EXISTS senior_operator;
DROP ROLE IF EXISTS operator        ;
DROP ROLE IF EXISTS operator_na     ;

CREATE ROLE bob   WITH LOGIN = true AND PASSWORD = 'password';
GRANT EXECUTE on INTERNAL SCHEME to bob  ;
CREATE ROLE nancy WITH LOGIN = true AND PASSWORD = 'password';
GRANT EXECUTE on INTERNAL SCHEME to nancy;
CREATE ROLE dirk  WITH LOGIN = true AND PASSWORD = 'password';
GRANT EXECUTE on INTERNAL SCHEME to dirk ;

CREATE ROLE senior_operator;
   //
   // INSERT, UPDATE, DELETE and TRUNCATE rows in any table in the specified
keyspace.
   //
GRANT MODIFY, SELECT ON KEYSPACE ks_6221 TO senior_operator;
GRANT senior_operator to nancy;

CREATE ROLE operator;
```

```
   //
GRANT MODIFY,SELECT ON TABLE ks_6221.cust_orders   TO operator;
GRANT SELECT        ON TABLE ks_6221.cust_payments TO operator;
   //
GRANT operator to bob;

CREATE ROLE operator_na;

// RESTRICT ROWS ON ks_6221.cust_orders USING other;
// InvalidRequest: Error from server: code=2200 [Invalid query]
message="Restrict Rows Statement must be for a Primary Key or a Partition Key
column"

RESTRICT ROWS ON ks_6221.cust_orders USING region;

GRANT SELECT ON 'NA'    ROWS IN ks_6221.cust_orders TO operator_na;
GRANT SELECT ON 'SEARS' ROWS IN ks_6221.cust_orders TO operator_na;
GRANT operator_na to dirk;
```

Relative to Example 20-2, the following is offered:

– The code above begins by creating 6 roles; 3 after named users, 3 virtual roles (not real persons) to assign to users.
– Passwords are assigned to the real users.
– INSERT, UPDATE, DELETE, and TRUNCATE are given via the MODIFY authorization. This and SELECT is given to `senior_operator` at the named keyspace level, of which nancy is a member.
– The role titled, operator, is given MODIFY and SELECT on one table, and just select on another table.
– Bob is granted operator authorization.
– `operator_na` is associated with row level access control; the RESTRICT ROWS command. And Dirk is given this level of authorization.

20.2.7 Test the Authentication/Authorization Above

Using the following code, confirm or deny what Bob, Dirk, and Nancy can perform-

```
cqlsh -u nancy -p password
cqlsh -u bob   -p password
cqlsh -u dirk  -p password
```

```
//  For each user above
use ks_6221;

select * from cust_orders;
select * from cust_payments;

select * from cust_orders where ;
select * from cust_payments;
```

20.3 In this document, we reviewed or created:

This month and in this document we detailed the following:

- How to implement table and row level security using DataStax Enterprise (DSE).
- How to disable the default named, cassandra user.
- And completed a high level overview of the entire security topic relative to DSE.

Persons who help this month.

Kiyu Gabriel, Matt Atwater, and Jim Hatcher.

Additional resources:

Free DataStax Enterprise training courses,

`https://academy.datastax.com/courses/`

Take any class, any time, for free. If you complete every class on DataStax Academy, you will actually have achieved a pretty good mastery of DataStax Enterprise, Apache Spark, Apache Solr, Apache TinkerPop, and even some programming.

This document is located here,

```
https://github.com/farrell0/DataStax-Developers-Notebook
https://tinyurl.com/ddn3000
```