Using MapR ACEs with Apache Drill to Control Access to Data

Document created by **cmatta** on May 4, 2016 • Last modified by **maprcommunity** on May 13, 2016
Version 11

Using MapR ACEs with Apache Drill to Control Data Access

MapR's unique ability to leverage Access Control Expressions (ACEs) on files and directories allows for fine-grained control over how data can be made available to downstream consumers. Combining this feature with Apache Drill's user impersonation functionality results in cell-level access control to heterogeneous data. Access to data cells can further be protected using Drill's UDF functionality to tokenize or mask the data. In this blog post I would like to illustrate how these technologies working together allow for cell-level access control to sensitive data and provide administrators peace of mind about who has access to what data.

MapR Access Control Expressions

From the MapR Documentation:

> Access Control Expressions (ACEs) allow you to define whitelists (to grant access) and blacklists (to deny access) for a combination of users, roles, and groups. You can grant different permissions to multiple users, groups, and roles for files, directories, and whole volume data using boolean expressions and subexpressions.

You can set ACEs using the `mfs` command: `hadoop mfs -setace`, you can read aces using `hadoop mfs -getace`.

Files have the following permissions:

Command Line Flag	Description
-readfile	Read a file.

Command Line Flag	Description
-writefile	Write to a file.
-executefile	Execute a file.

Directories have these permissions:

Command Line Flag	Description
-readfile	Read a file.
-writefile	Write to a file.
-executefile	Execute a file.
-readdir	List the contents of a directory. This access is required to write and/or execute files in the directory.
-lookupdir	Lookup a file in a directory. This access is required to find, read, write, and/or execute files in the directory.
-addchild	Add a file or subdirectory.
-deletechild	Delete a file or subdirectory.

The directory ACE is inherrited by any files created within that directory, existing files and directories remain unchanged.

When a file or directory has an ACE set the mode line indicates that an ACE is present with a plus when using the `hadoop fs -ls` command:

```
$ hadoop fs -ls -d /data/userdata
d-wx------+ - mapr mapr 8 2016-04-29 20:48 /data/userdata
```

Demo

So let's say we have some sensitive data about our users that's stored as JSON and it looks like this:

```
{
  "id": 6,
  "name": "Keith Patterson",
  "address": "16019 Strong Mill Chase",
  "gender": "MALE",
  "zip": {
  "zip": "15015",
  "city": "BRADFORDWOODS",
  "latitude": "40.63",
  "decommisioned": "false",
  "locationType": "PRIMARY",
  "location": "NA-US-PA-BRADFORDWOODS",
  "estimatedPopulation": "1237",
  "state": "PA",
  "zipType": "STANDARD",
  "taxReturnsFiled": "676",
  "totalWages": "49171202",
  "longitude": "-80.08"
  },
  "ssn": {
  "state": "NV",
  "ssn": "530-98-7595"
  }
}
```

note: this synth data was generated using the excellent log-synth using this schema.

We can see some sensitive information in this data, chiefly the SSN. Drill and MapR ACEs allow for a subset of this data to be made visible to certain users without compromising access to the whole data set. We can use ACEs to lock down access to the source data (note the use of -R flag for recursively setting the ACE on all files and directories below the target):

```
  $ hadoop mfs -setace -R \
    -readfile 'u:mapr' \
    -writefile 'u:mapr' \
    -executefile 'u:mapr' \
    -lookupdir 'u:mapr' \
    -addchild 'u:mapr' \
    -deletechild 'u:mapr' /data/userdata
```

In order to allow access to certain pieces of this data we can ensure Drill impersonation is enabled by following the directionshere.

Let's create a view into this data that excludes certain sensitive fields, maybe our analysts only need to see the id, Name, Gender and Zip for this data:

```
0: jdbc:drill:> CREATE OR REPLACE VIEW dfs.data.`users` as SELECT
. . . . . . . > CAST(x.`id` as BIGINT) as `id`,
. . . . . . . > CAST(x.`name` as VARCHAR(255)) as `name`,
. . . . . . . > CAST(x.`gender` as VARCHAR(64)) as `gender`,
. . . . . . . > CAST(x.`zip`.`zip` as INT) as `zipcode`
. . . . . . . > FROM dfs.data.`userdata` as x;
+-------+----------------------------------------------------------+
| ok | summary |
+-------+----------------------------------------------------------+
| true | View 'users' created successfully in 'dfs.data' schema |
+-------+----------------------------------------------------------+
1 row selected (0.141 seconds)
```

This creates a `users.view.drill` file in the root of our `data` workspace (defined in the dfs Drill storage plugin). This file can be copied into the user directories where it will inherit the ACEs applied to those directories or stored in a central location and multiple users can be granted access to the one file (this is run as the mapr user):

```
$ hadoop mfs -setace -readfile 'u:mapr|u:cmatta' /data/users.view.drill
```

Now select it with Drill (this is run as the `cmatta` user):

```
0: jdbc:drill:> select * from dfs.data.users limit 5;
+-----+-------------------+--------+----------+
| id | name | gender | zipcode |
+-----+-------------------+--------+----------+
| 5 | Lillian Aaron | FEMALE | 30582 |
| 31 | Sherri Owens | FEMALE | 43162 |
| 33 | George Villatoro | MALE | 36870 |
| 36 | Sandra Heidrick | FEMALE | 54960 |
| 39 | Larry Adkins | FEMALE | 40475 |
+-----+-------------------+--------+----------+
5 rows selected (0.752 seconds)
```

But the user `cmatta` can't read the original source data:

```
0: jdbc:drill:> select * from dfs.data.userdata limit 5;
Error: PERMISSION ERROR: Not authorized to read table [userdata] in schema [dfs.data]
```

```
SQL Query null

[Error Id: f82b5f4a-947c-435e-ad29-77ce28dd3e43 on ip-172-16-1-221.ec2.internal:31010]
(state=,code=0)
```

Incorporating User Defined Functions for data masking

Maybe we have downstream consumers that require the SSN field, but we don't want to expose our customer's PII. We can extend Drill with a user defined function to mask the data. I'm using the mask function found in this simple Drill functions repository.

After building the package and putting the resulting jars in `/opt/mapr/drill/drill-1.6.0/jars/3rdparty` on each Drill node we restart all Drillbits on the cluster.

Now we can create a new view:

```
CREATE OR REPLACE VIEW dfs.data.`users_ssns` as SELECT
CAST(x.`id` as BIGINT) as `id`,
CAST(x.`name` as VARCHAR(255)) as `name`,
CAST(x.`gender` as VARCHAR(64)) as `gender`,
CAST(x.`zip`.`zip` as INT) as `zipcode`,
MASK(CAST(x.`ssn`.`ssn` as VARCHAR(12)), '0', 3) as `masked_ssn`,
CAST(x.`ssn`.`state` as VARCHAR(2)) as `ssn_issuing_state`
FROM dfs.data.`userdata` as x;
```

And set the ACE on the new view so that our analysts that need the masked SSNs can read the data:

```
[mapr@ip-172-16-1-220 ~]$ hadoop mfs -setace -writefile 'u:mapr' -readfile
'u:mapr|g:analysts' -executefile 'u:mapr' /data/users_ssns.view.drill
[mapr@ip-172-16-1-220 ~]$ hadoop mfs -getace /data/users_ssns.view.drill
Path: /data/users_ssns.view.drill
  readfile: u:mapr | g:analysts
  writefile: u:mapr
  executefile: u:mapr
  mode: rwxr-----
```

and the resulting query shows that our SSNs are available but masked, this is run as the cmatta user who is in the analysts group:

```
0: jdbc:drill:> select * from dfs.data.users_ssns limit 10;
+-----+--------------------+---------+----------+--------------+-------------------+
| id | name | gender | zipcode | masked_ssn | ssn_issuing_state |
+-----+--------------------+---------+----------+--------------+-------------------+
| 5 | Lillian Aaron | FEMALE | 30582 | 000-03-5398 | MO |
| 31 | Sherri Owens | FEMALE | 43162 | 000-79-9530 | AZ |
| 33 | George Villatoro | MALE | 36870 | 000-47-5807 | TX |
| 36 | Sandra Heidrick | FEMALE | 54960 | 000-42-4139 | OK |
| 39 | Larry Adkins | FEMALE | 40475 | 000-24-6091 | MS |
| 42 | Verna Mcdonald | FEMALE | 67851 | 000-63-7130 | WY |
| 45 | Rick Burden | FEMALE | 19902 | 000-93-6014 | DC |
| 48 | Kathleen Correia | FEMALE | 50831 | 000-36-9646 | CT |
| 51 | Joan Westmoreland | FEMALE | 47035 | 000-92-7688 | WY |
| 54 | Frank Scott | MALE | 97238 | 000-41-6338 | AK |
+-----+--------------------+---------+----------+--------------+-------------------+
10 rows selected (0.405 seconds)
```

Just to prove it's working, let's try and access the data as `user01` who is not a member of the analysts group:

```
[user01@ip-172-16-1-220 ~]$ id
uid=5602(user01) gid=5602(user01) groups=5602(user01)
[user01@ip-172-16-1-220 ~]$ sqlline -u jdbc:drill: -n user01
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=512M; support was removed in
8.0
apache drill 1.6.0
"what ever the mind of man can conceive and believe, drill can query"
0: jdbc:drill:> use dfs.data;
+-------+--------------------------------------+
| ok | summary |
+-------+--------------------------------------+
| true | Default schema changed to [dfs.data] |
+-------+--------------------------------------+
1 row selected (0.239 seconds)
0: jdbc:drill:> show tables;
+---------------+--------------+
| TABLE_SCHEMA | TABLE_NAME |
+---------------+--------------+
+---------------+--------------+
No rows selected (0.171 seconds)
0: jdbc:drill:> select * from dfs.data.users_ssns;
Error: VALIDATION ERROR: From line 1, column 15 to line 1, column 17: Table
'dfs.data.users_ssns' not found

SQL Query null

[Error Id: e73bb639-d760-4af5-bea1-5d7e1c4e2f64 on ip-172-16-1-221.ec2.internal:31010]
(state=,code=0)
```

Notice that `user01` can't even see that a `users_ssns` view even exists let alone run a query against the view.

Hopefully this was illustrative of the power and flexibility that Drill and MapR ACEs bring to SQL on Hadoop.

ATTACHMENTS

Visibility: Answers • **1421 Views**

Last Modified by **maprcommunity** on May 13, 2016 8:29 AM

Tags: security drill knowledge article apache drill sql mapr-fs mapr security

Categories: Analytics Security

0

0 Comments

Recommended Content

- MapR 6.0 Install Failed
- MapR 6.0 Sandbox
- Connecting to MapR Cluster using Java API
- Lab to practice hands-on
- DEV 350 Lesson 2 Is never marked as finished

Incoming Links

- Mid-Ohio Hadoop User Group - August 30, 2016 - OH

Home | Top of page | Help