MongoDB Data Security - Custom Roles and Views

Room Texas 6 - 16:10
Adamo Tonete

I've been working at Percona since 2015 as a Senior Support Engineer.
Agenda

- Installing MongoDB in a secure way
- Default roles
- Creating your own role
- Using views
- Views + User Defined Roles for best security
- Questions
Installing MongoDB

By default MongoDB doesn't come with authentication and for this reason we do see a lot of news reporting data leaks.

From version 4.0+ it is mandatory to set the bindIP, or specify manually if the database must listen to all IPS.
For new versions it is necessary to set a listening IP, which means the database will only answer queries and commands which come from this IP address.
Installing MongoDB - Listen IP

Bad Practice

```yaml
net:
  bindIp: 0.0.0.0
```

Good Practice

```yaml
net:
  bindIp: 172.10.10.122
```
Installing MongoDB - Enabling Authentication

Authentication is not enabled by default, we need to configure and create the root user as the first step for a secure environment.
Installing MongoDB - Enabling Authentication

```javascript
mongod.conf
  authorization.enabled : true

use admin
db.createUser({
  user : 'administrator',
  pwd : '123321',
  roles : ['root'])
```
The minimum security option for a replica set is having a key file, that will ensure the instances can talk each other.
Installing MongoDB - Replicsets?

openssl rand -base64 756 > mykeyfile
chmod 400 mykeyfile

mongod.conf
security.keyFile : mykeyfile

Alert: This change enables authentication as well!
Still talking about new versions, new users can have an IP number and the database will only accept commands from there.
Authentication Restrictions

use admin

db.createUser({
  user: 'local_administrator',
  pwd: '123321',
  roles: ['root'],
  authenticationRestrictions: {
    clientSource: ['127.0.0.1']
  }
})
Database comes with several roles - that is enough for most of the cases
All the roles listed below come by default in the MongoDB database server

<table>
<thead>
<tr>
<th>read</th>
<th>readWrite</th>
<th>dbAdmin</th>
<th>dbOwner</th>
<th>userAdmin</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterAdmin</td>
<td>clusterManager</td>
<td>clusterMonitor</td>
<td>hostManager</td>
<td>backup</td>
</tr>
<tr>
<td>restore</td>
<td>readAnyDatabase</td>
<td>readWriteAnyDatabase</td>
<td>userAdminAnyDatabase</td>
<td></td>
</tr>
<tr>
<td>dbAdminAnyDatabase</td>
<td>root</td>
<td>__system</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
use admin

db.createUser({user : 'read_any',
    pwd : '123',
    roles : ['readAnyDatabase']})
Creating Custom Role

db.createRole({
    role: "view_employee",
    privileges: [
        { resource: { db: "percona", collection: "employees" }, actions: ["find","collStats"]}
    ],
    roles: [
        { role: "read", db: "admin" }
    ]
})
How to create and maintain a view
Views

Views are pre-established code that is executed when querying from them.

For a user a view is just a collection and by default a view is read only. Views can run simple queries or complex aggregation pipelines.

For this example we are going to create a view that only gives employee name and id to a third party provider that will integrate with us.
Creating a View

Use database

db.createView('vw_emp_names', 'employee',
    [{ $project: { _id: 1, name: 1 } } ]
)
Creating View

How to create a view?

From the docs:
```javascript
db.createView(<view>, <source>, <pipeline>, <options>)
```

Options is basically the collation
```
collation: {
  locale: <string>,
  caseLevel: <boolean>,
  caseFirst: <string>,
  strength: <int>,
  numericOrdering: <boolean>,
  alternate: <string>,
  maxVariable: <string>,
  backwards: <boolean>
}
```
Acceptable Pipeline Operator

All the operators used in a aggregation are available in a view meaning you can use $match, $unwind, $project.. and so on..

https://docs.mongodb.com/manual/meta/aggregation-quick-reference/
Accessing a view

In order to execute the view code we need to invoke a find command

The following command executes the code:

```javascript
   db.vw_emp_names.find()
```

Views are also visible as a collection, a `show collections` command will return the views as well.
How to control who can query a view
use admin

db.createRole(
  {
    role: "view_views",
    privileges: [
      { resource: { db: "percona", collection: "system.views" }, actions: [ "find" ] },
      { resource: { db: "percona", collection: "employees_name" }, actions: [ "find","collStats"] }
    ],
    roles: [
      { role: "read", db: "admin" }
    ]
  }
)
use admin

db.createUser({user: 'intern', pwd: '123', roles: ['view_views']})
Live Demonstration
Live Demonstration

<live demo>
Questions
Thank You to Our Sponsors
Rate My Session