MODUL 8
SECURITY MONGODB

8.1. Tujuan Praktikum

1. Mahasiswa mampu memahami security pada MongoDB
2. Mahasiswa mampu mengimplementasikan Authentication pada MongoDB
3. Mahasiswa mampu mengimplementasikan Authorization pada MongDB

8.2. Dasar Teori

8.2.1. Security MongoDB

MongoDB menyediakan berbagai fitur, seperti autentikasi, kontrol akses, enkripsi, untuk mengamankan penerapan MongoDB. Beberapa fitur keamanan utama meliputi:

1. Authentication
2. Authorization
3. TLS/SSL

8.2.2. Authentication

Authentication adalah proses memverifikasi identitas klien. Ketika kontrol akses (otorisasi) diaktifkan, MongoDB mengharuskan semua klien untuk mengautentikasi diri mereka sendiri untuk menentukan akses mereka.

Meskipun otentikasi dan otorisasi berhubungan erat, otentikasi berbeda dengan otorisasi:

1. Otentikasi memverifikasi identitas pengguna.
2. Otorisasi menentukan akses pengguna terverifikasi ke sumber daya dan operasi.

MongoDB punya beberapa mekanisme Authentication, diantaranya:

1. *SCRAM Authentication*

   Mekanisme Autentikasi Salted Challenge Response Authentication Mechanism (SCRAM) adalah mekanisme autentikasi default untuk MongoDB.

2. *x.509 Certificate Authentication*

   MongoDB mendukung autentikasi sertifikat x.509 untuk autentikasi klien dan autentikasi internal anggota set replika dan cluster yang dipecah. Autentikasi sertifikat x.509 memerlukan koneksi TLS/SSL yang aman.

Untuk menggunakan MongoDB dengan x.509, diwajibkan untuk menggunakan sertifikat valid yang dibuat dan ditandatangani oleh otoritas sertifikat. Sertifikat x.509 klien harus memenuhi persyaratan sertifikat klien.

3. *Kerberos Authentication*

   MongoDB Enterprise mendukung Autentikasi Kerberos. Kerberos adalah protokol otentikasi standar industri untuk sistem klien/server besar yang menyediakan otentikasi menggunakan token berumur pendek yang disebut tiket.

   Untuk menggunakan MongoDB dengan Kerberos, harus memiliki penerapan Kerberos yang dikonfigurasi dengan benar, prinsipal layanan Kerberos yang dikonfigurasi untuk MongoDB, dan prinsipal pengguna Kerberos yang ditambahkan ke MongoDB.

4. *LDAP Proxy Authentication*

   MongoDB Enterprise dan MongoDB Atlas mendukung autentikasi *proxy* LDAP Proxy Authentication melalui layanan *Lightweight Directory Access Protocol* (LDAP).

5. *Internal / Membership Authentication*

   Selain memverifikasi identitas klien, MongoDB dapat meminta anggota set replika dan kluster terpecah untuk mengautentikasi keanggotaan mereka pada set replika atau kluster terpecah masing-masing. Lihat Otentikasi Internal/Keanggotaan untuk informasi lebih lanjut.

8.2.3. Authorization

Authorization pada MongoDB adalah mekanisme yang digunakan untuk mengontrol akses pengguna ke database dan koleksi data tertentu dalam lingkungan MongoDB. Dengan menggunakan authorization, administrator database dapat menentukan hak akses dan peran untuk setiap pengguna atau peran yang terkait dengan basis data.

MongoDB menyediakan beberapa metode untuk melakukan authorization:

1. *Built-in Role-based Access Control (RBAC):* MongoDB memiliki beberapa peran bawaan yang dapat diberikan kepada pengguna, seperti read, readWrite, dbAdmin, dbOwner, dll. Setiap peran memiliki hak akses yang ditentukan terhadap basis data dan koleksi yang berbeda.

Roles	Hak
Read	Membaca data pada database

Write	Memiliki semua hak dari Role Read dan memiliki hak untuk menambah dan mengubah data.
dbAdmin	Memiliki hak untuk melakukan perintah administrative seperti indexing, perintah yang berhubungan dengan schema.
userAdmin	Memiliki hak untuk membuat dan memodifikasi roles dan user pada database yang ditentukan.
dbOwner	Memiliki hak gabungin dari Role ReadWrite, userAdmin, dbAdmin.
readAnyDatabase	Memiliki hak yang sama dengan Role Read namun berlaku pada semua database keculai local dan config.
writeAnyDatabase	Memiliki hak yang sama dengan Role Write namun berlaku pada semua database keculai local dan config.
userAdminAnyDatabase	Memiliki hak yang sama dengan Role userAdmin namun berlaku pada semua database keculai local dan config.
dbAdminAnyDatabase	Memiliki hak yang sama dengan Role dbAdmin namun berlaku pada semua database keculai local dan config
Root	Memiliki hak akses dari semua Role.

2. *Custom Roles:* Selain peran bawaan, administrator database juga dapat membuat peran kustom dengan hak akses yang disesuaikan sesuai dengan kebutuhan aplikasi.
3. *Authentication Mechanisms:* Sebelum melakukan authorization, MongoDB memerlukan proses autentikasi pengguna. MongoDB mendukung beberapa mekanisme autentikasi, seperti username/password, Kerberos, LDAP, dan X.509 certificate authentication.
4. *Access Control Lists (ACLs):* Administrator database dapat mengatur kontrol akses lebih lanjut menggunakan daftar kendali akses (ACLs) yang memungkinkan pengaturan akses yang lebih rinci pada level koleksi dan dokumen individu.

Dengan menggunakan authorization, administrator database dapat membatasi akses pengguna hanya ke data yang relevan, menjaga keamanan dan integritas data, serta melindungi basis data dari akses yang tidak sah. Penting untuk mengatur authorization dengan hati-hati dan memberikan hak akses yang sesuai kepada pengguna sesuai dengan tanggung jawab dan kebutuhan aplikasi.

8.2.4. TLS/SSL (Transport Encryption)

MongoDB mendukung TLS/SSL (Transport Layer Security/Secure Sockets Layer) untuk mengenkripsi semua lalu lintas jaringan MongoDB. TLS/SSL memastikan bahwa lalu lintas jaringan MongoDB hanya dapat dibaca oleh klien yang dituju. MongoDB menonaktifkan dukungan untuk enkripsi TLS 1.0 pada sistem yang mendukung TLS 1.1+. Untuk detail lebih lanjut, lihat Nonaktifkan TLS 1.0.

MongoDB menggunakan pustaka native TLS/SSL OS libraries:

Platform	TLS/SSL Library
Windows	Secure Channel (Schannel)
Linux/BSD	OpenSSL
macOS	Secure Transport

Gambar 21. Pustaka native TLS/SSL OS

TLS/SSL Ciphers. Enkripsi TLS/SSL MongoDB hanya mengizinkan penggunaan cipher TLS/SSL yang kuat dengan panjang kunci minimal 128-bit untuk semua koneksi.

Certificates. Untuk menggunakan TLS/SSL dengan MongoDB, diwajibkan untuk memiliki sertifikat TLS/SSL sebagai file PEM, yang merupakan wadah sertifikat yang digabungkan. MongoDB dapat menggunakan sertifikat TLS/SSL yang valid yang diterbitkan

oleh otoritas sertifikat atau sertifikat yang ditandatangani sendiri. Untuk penggunaan produksi, penerapan MongoDB harus menggunakan sertifikat valid yang dibuat dan ditandatangani oleh otoritas sertifikat yang sama. Dapat juga membuat dan mengelola otoritas sertifikat independen, atau menggunakan sertifikat yang dibuat oleh vendor TLS/SSL pihak ketiga. Menggunakan sertifikat yang ditandatangani oleh otoritas sertifikat tepercaya memungkinkan driver MongoDB memverifikasi identitas server.

8.3. *Software*

1. MongoDB
2. Mongo shell

8.4. Tahapan Kerja

8.4.1. Authentication MongoDB

1. Memulai MongoDB tanpa autentikasi
2. Hubungkan ke server menggunakan perintah mongosh
3. Membuat administrator pengguna

   > use **admin**

   Tahap pertama adalah untuk membuat user dengan role **root**, yang memberikan hak istimewa semua role menjadi satu. Contoh berikut ini akan membuat pengguna mongo dengan kata sandi **"mongo":**

```
db.createUser(
    {
        user: "mongo",
        pwd: "mongo",
        roles: [
            "root",
        ]
    }
)
```

   Kemudian putuskan sambungan dari shell mongo **(Ctrl+D)**.

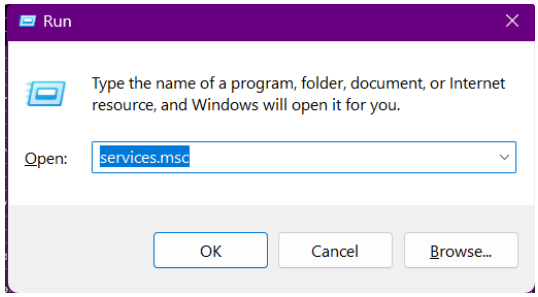
4. Mengaktifkan autentikasi dalam berkas konfigurasi mongod.

   Buka “C:\Program Files\MongoDB\Server\6.0\bin\mongod.cfg” dengan kode editor dan tambahkan baris berikut:

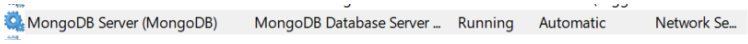
```
Security:
  Authorization: "enabled"
```

```
security:
  authorization: "enabled"
```

   Setelah selesai buka Microsoft service dengan menekan tombol Start + R lalu ketik “services.msc”.



Kemudian cari service dengan nama “MongoDB Server”.



Klik kanan pada service tersebut lalu pilih “restart”.

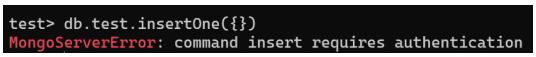
Mulai sekarang, semua klien yang terhubung ke server ini harus mengautentikasi diri mereka sendiri sebagai pengguna yang valid, dan mereka hanya akan dapat melakukan tindakan sebagaimana ditentukan oleh peran yang ditetapkan.

5. Login sebagai root bisa dilakukan dengan menggunakan perintah mongosh namu menambahkan beberapa argument menjadi : “mongosh --username root --password root.

```
C:\Users\Denny->mogosh –username mongo –-password mongo
```

6. Apabila kita masuk ke terminal mongosh tanpa username dan password kita tidak akan bisa menjalankan perintah untuk berinteraksi dengan database dikarenakan kita tidak memiliki Authentication.

```
Test> db.test.insertOne({})
```



8.4.2. Authorization MongoDB

Authorization merupakan proses yang dilakukan setelah Authentication berhasil, proses ini dilakukan untuk memeriksa Apakah user memiliki hak akses untuk melakukan sebuah aksi. Hak akses ini disimpan dalam bentuk “Role”.

1. Buat pengguna lain yang memiliki role selain root.

   Login menggunakan akun root lalu jalankan perintah berikut ini menambahkan pengguna contoh ke basis data rentalfilm yang memiliki peran read dalam basis data rentalfilm:

```
test> use rentalfilm
switched to db rentalfilm
rentalfilm > db.createUser(
       {
          user: "contoh",
          pwd: "contoh",
          roles: [
              { role: "read", db: "rentalfilm" }
          ]
       }
);
```

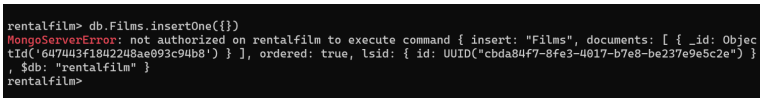
Perintah diatas akan membuat user baru pada basis data rentalfilm yang memiliki role "read", artinya user tersebut bisa melihat database namun tidak bisa melakukan perintah lain seperti insert, update, dll.

2. Login Sebagai user selain root memiliki perintah yang sedikit berbeda contohnya sebagai berikut:

```
C:\Users\Denny->mogosh --username contoh --password contoh –
authenticationDatabase rentalfilm
```

3. Ketika menggunakan user yang hanya memiliki role "read" dan menjalankan perintah insert maka akan mendapatkan error.

```
rentalfilm> db.Films.insertOne({})
```



4. Buat pengguna baru dengan role "readWrite".

```
rentalfilm > db.createUser(
       {
          user: "conto2h2",
          pwd: "contoh",
          roles: [
              { role: "readWrite", db: "rentalfilm" }
          ]
       }
);
```

```
rentalfilm> db.createUser(
...     {
...         user: "contoh2",
...         pwd: "contoh2",
...         roles: [
...             { role: "readWrite", db: "rentalfilm" }
...         ]
...     }
... );
{ ok: 1 }
```

Perintah diatas akan membuat pengguna dengan username contoh2 dan password contoh2 yang memiliki role readWrite yaitu membaca dan menulis pada database rentalfilm.

5. Login menggunakan user contoh2 dan jalankan perintah insertOne.

```
C:\Users\Denny->mogosh --username contoh2 --password contoh2 --
authenticationDatabase rentalfilm
```

```
Rentalfilm> db.test.insertOne({});
```

```
C:\Users\Denny>mongosh --username contoh2 --password contoh2 --authenticationDatabase rentalfilm
```

```
rentalfilm> db.test.insertOne({});
{
  acknowledged: true,
  insertedId: ObjectId("6474475e66952f81c5bf5e63")
}
```

Apabila user memiliki authorization yang mengizinkan user tersebut untuk melakukan perintah insert maka, query akan sukses dijalankan.

6. Di MongoDB mendukung pembuatan custom role yang bisa memiliki hak yang berbeda. Contohnya ketika ingin membatasi pengguna hanya bisa mengakses 1 buah collection dalam sebuah database kita bisa menggunakan “Privileges”. Privileges merupakan cara untuk membatasi akses dalam level Collections.

   Untuk membuat role baru, wajib memiliki user dengan role “userAdmin”, “dbAdmin” atau “root”. Perintah yang digunakan untuk membuat role baru adalah :

```
rentalfilm> db.createRole({
        role: "readOnly", //nama role
        privileges: [], //privileges
        roles: [        //daftar array role untuk mengambil privileges dari
role yang sudah ada
                { role: "read", db: "rentalfilm" }
           ]
       });
```

```
rentalfilm> db.createRole({
...     role: "readOnly", //nama role
...     privileges: [], //privileges
...     roles: [        //daftar array role untuk mengambil privileges dari role yang sudah ada
...         {
...             role: "read",
...             db: "rentalfilm"
...         }
...     ]
... });
```

Query diatas akan membuat role baru dengan nama readOnly yang hanya memiliki hak untuk membaca data di database rentalfilm.

7. Kita bisa melihat daftar roles yang ditambahkan dengan menggunakan perintah db.getRoles().

```
rentalfilm> db.getRoles();
```

8. Kita bisa menggubah salah satu user yang sudah ada dan mengganti role nya dengan role yang baru saja dibuat dengan menggunakan perintah updateUser.

```
rentalfilm> db.updateUser("contoh2",
       {
          Roles: [
              { role: "readOnly", db: "rentalfilm" }
          ]
       }
);
```

Ketika user contoh2 melakukan perintah insert maka akan muncul error:

```
rentalfilm> db.city.insertOne({});
```

9. Untuk membatasi hak akses collection kita kisa menambahkan di bagian Privileges ketika membuat role atau melakukan update pada role yang sudah dibuat dengan menggunakan perintah updateRole.

```
rentalfilm> db.updateRole("readOnly", {
        privileges: [
            {
              resource: {
                 db: "rentalfilm", //nama database
                 collection: "city" // nama collection yang diberi akses
              },
       ],
        roles: [
              { role: "read",
                db: "rentalfilm"
              }
          ]
   });
```

```
rentalfilm> db.updateRole("readOnly", {
...     privileges: [
...         {
...             resource: {
...                 db: "rentalfilm",  //nama database
...                 collection: "city" // nama collection yang diberi akses
...             },
...             actions: [ "insert" ]  //query yang bisa dijalankan
...         }
...     ],
...     roles: [
...         {
...             role: "read",
...             db: "rentalfilm"
...         }
...     ]
... });
{ ok: 1 }
```

10. Login kembali menggunakan user contoh2 dan jalankan perintah insert data.

```
rentalfilm> db.city.insertOne({});
```

```
rentalfilm> db.city.insertOne({});
{
  acknowledged: true,
  insertedId: ObjectId("64744cb609e50ed5bd12c6d0")
}
```

Karena role readOnly sudah ditambahkan privileges untuk melakukan insert maka query tersebut berhasil berjalan.

8.5. Tugas dan Latihan

1. Buat user baru dengan role readWrite dan jalankan query untuk menambahkan lalu melihat data pada database rentalfilm.
2. Hapus user baru yang sudah dibuat.
3. Buat role baru yang hanya memiliki hak untuk membaca semua collection dan menghapus data dari sebuah collection.
4. Buat user baru dengan role yang baru dibuat dari nomor 3 dan jalankan perintah find, insert, delete dan update.