MongoDB 보안 인증

mongoDB 인증 방식

SCRAM

Salted Challenge Response Authentication Mechanism 은 mongodb 의 기본 인증으로 사용자 인증에 대한 표준인 IETF RFC 5802 를 구현한 인증이다.

mongodb 는 SCRAM 을 사용하여 사용자 username, password, authentication database 에 대한 사용자 자격 인증을 확인 한다.

x.509

mongodb 는 클라이언트 인증 과 replicaset 및 cluster member 간의 인증을 위해 x.509 를 사용 한다. x.509 인증을 위해 TLS/SSL 이 요구된다.

프로덕션 운영을 위해 특정 기관에서 생성 및 관리되는 인증서를 사용해야 한다. 클라인트에서 x.509 를 사용하는 경우 id/passwd 없이 인증할 수 있다.

Kerberos Authentication

mongodb enterprise 에서 제공되는 인증 방식이다. Kerberos Authentication 클라이언트에서 mongod 및 mongos 로의 인증에서 사용된다.

Kerberos Authentication 는 대규모 산업 표준 인증 프로토롤로서 MIT implentation 만 지원한다. 그 만큼 대중적이고 많이 쓰이기 때문에 기존에 이를 채택하고 있다면 큰 이점이 될 수 있다.

LDAP Proxy Authentication

mongodb enterprise 에서 제공되는 인증 방식이다. LDAP 은 x.500 DAP 가 경량화된 Lightweight DAP 이다. mongodb 에서는 다음 두가지 방식을 지원한다.

via	**desc**
os library	- 3.4 부터 os library 기반의 LDAP 을 지원 - linux 및 windows 모두 지원
saslauthd	saslauthd 를 통한 LDAP 은 linux 에서만 지원

내부 인증

replicaset 및 cluster member 간의 내부 인증에서는 keyFile 과 x.509 로 인증을 할 수 있다. 개발 에서는 keyFile 인증을 사용하지만 실제 운영에서는 x.509 방식을 추천한다.

사용자 인증

mongodb 는 일반 DBMS 와 마찬가지로 id/pass 방식의 사용자 인증을 내부적으로 지원하며 여기에 인증 데이터베이스 정보가 추가로 지정된다. 인증 데이터베이스는 특정 데이터베이스로 이동하여 사용자를 생성할때 그

당시의 데이터베이스가 지정된다. 물론 한 사용자에 복수의 데이터베이스에 대한 role 을 부여할 수도 있다.

mongodb 에서 제공하는 built-in-role 은 다음과 같다.

category	**role**	**desc**
database user roles		
	read	모든 비시스템 캘럭션 및 system.js 컬렉션에 대한 읽기 권한
	readWrite	read 에 쓰기가 더해진 권한
database admin roles		
	dbAdmin	특정 데이터베이스의 스키마 관련 업무, 인덱싱, 통계수집에 대한 권한
	userAdmin	특정 데이터베이스의 롤과 사용자에 대한 생성 및 관리 권한
	dbOwner	특정 데이터베이스의 readWrite, dbAdmin, userAdmin 을 하나로 모은 권한
cluster administration role		
	clusterManager	cluster 에 대한 관리 및 모니터링 권한, config 및 local 데이터베이스 에도 억세스 할 수 있다.
	clusterMonitor	monogdb cloud manager, ops manager 같은 읽기 전용 모니터링 툴에 사용되는 권한
	hostManager	서버들에 대한 관리 및 모니터 권한
	clusterAdmin	clusterManager, ClusterMonitor, hostManager 를 하나로 모은 권한
backup and restoration roles		
	backup	monogdb cloud manager, ops manager, mongodump 등에서 사용되며 mongod 인스턴스를 백업하기 위해 필요한 권한
	restore	mongorestore 등에서 백업된 데이터를 대상으로 복구를 수행하는데 필요한 권한
all-database roles		
	readAnyDatabase	local 및 config 를 제외한 모든 데이터베이스에 대한 읽기 권한

category	role	desc
	readWrieteAnyDatabase	readAnyDatabase 에 쓰기가 더해진 권한
	userAdminAnyDatabase	local 및 config 를 제외한 모든 데이터베이스에 대해 사용자 관리 권한
	dbAdminAnyDatabase	local 및 config 를 제외한 모든 데이터베이스에 dbadmin 과 같은 권한
superuser roles		
	root	모든 자원에 대한 모든 권한
internal roles		
	__system	replicaset, cluster 멤버등에게 내부적으로 사용되는 권한으로 일반 사용자에게는 부여하지 않아야 한다.

모든 권한을 가진 사용자 생성

```
// super user 생성
> db.createUser( { user:"sysdba", pwd : "sysdba", roles: [ "root" ]} )
Successfully added user: { "user" : "sysdba", "roles" : [ "root" ] }

// 접속
$ bin/mongo -u "sysdba" -p "sysdba" --authenticationDatabase "admin"
MongoDB shell version v4.4.3
```

일반 데이터베이스 사용자 생성

```
> db.createUser( { user:"db1user", pwd :"db1user",  roles:[ "readWrite" ] } )
Successfully added user: { "user" : "db1user", "roles" : [ "readWrite" ] }

// db1 유저 db1user 에게 db2 권한을 부여 by root 유저
> use db1
switched to db db1
> db.grantRolesToUser(  "db1user", [ {role:"readWrite",db:"db2"} ] )

> use admin
> db.system.users.find().pretty()
... ...
{
        "_id" : "db1.db1user",
        "userId" : UUID("de051c0d-f25f-4367-b6a5-11a27452f4d6"),
        "user" : "db1user",
        "db" : "db1",
        "credentials" : {
                "SCRAM-SHA-1" : {
                        "iterationCount" : 10000,
                        "salt" : "L7SzlUA419LtcYhZKVEvhw==",
```

```
                        "storedKey" : "+c0uQMnIojGMxtQDWbj8rspu/Xc=",
                        "serverKey" : "MmZTPP2hJm9teseHQEkh6beXRV0="
                },
                "SCRAM-SHA-256" : {
                        "iterationCount" : 15000,
                        "salt" : "au6BMjW0Z+cMe6uoUsQkDZerILGD9nvsWlxCfg==",
                        "storedKey" :
"10AWnil+KOAd9LLTci6VuujebJsxEcFTWtCDeMlTGOE=",
                        "serverKey" :
"D7XEvOzQrSlFPUTBnNCel9NQ8YDXRh+Zk754TvNwY0M="
                }
        },
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "db1"
                },
                {
                        "role" : "readWrite",
                        "db" : "db2"
                }
        ]
}

// 주의점
// 특정 db 에 종속적인 계정을 만들때는 반드시 해당 db 로 이동해야 한다.
> use admin
switched to db admin
> db.createUser( { user:"db1user2", pwd :"db1user",  roles:[
{role:"readWrite",db:"db1"} ] } )
Successfully added user: {
        "user" : "db1user2",
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "db1"
                }
        ]
}
> db.system.users.find().pretty()
... ...
{
        "_id" : "admin.db1user2",
        "userId" : UUID("089ac595-09fa-4f36-b887-7b93e990d604"),
        "user" : "db1user2",
        "db" : "admin",
        "credentials" : {
                "SCRAM-SHA-1" : {
                        "iterationCount" : 10000,
                        "salt" : "cCDNOD5Ctt4SzWgLD05f8w==",
                        "storedKey" : "O2b1FHycp7iBhUKZyYjj09XLF5U=",
                        "serverKey" : "XzrRO3sGnv6lBCa8mTrX1uT7dwQ="
                },
                "SCRAM-SHA-256" : {
```

```
                        "iterationCount" : 15000,
                        "salt" : "JEIsFPJlPs0Mdm+GyYr3svbrABNkRcmwxF1UVw==",
                        "storedKey" :
"9kiU1yxLpXvDsDA0IQbO4/GKcCTZkTFEJIXkIUS2XKc=",
                        "serverKey" :
"LIkhx0AQ4PkPytFNntl+tCgno4ABPw1/2X64BIzufD8="
                }
        },
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "db1"
                }
        ]
}
// 생성시 db가 인증db가  된다.
$ bin/mongo -u "db1user2" -p "db1user" --authenticationDatabase "admin"
MongoDB shell version v4.4.3
connecting to: mongodb://127.0.0.1:27017/?
authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("68ec90b1-681a-4f0b-88f8-d16e04e0fe82") }
MongoDB server version: 4.4.3
>
```