

MongoDB教育訓練-20211223-04-關於安全

Topics we cover

- Introduction to encryption and PKI
- Authentication mechanisms
- Authorization
- Roles
- Encryption
 - In Flight
 - At Rest
 - In Use
- Auditing
- Additional security measures

對稱加密 && 非對稱加密

Encryption

- Encryption protects data using a cryptographic key
- Keys allow us to lock and unlock things
- Specifically to convert data between readable (plaintext) and unreadable (ciphertext)
- Two types of encryption:
 - Symmetric
 - Asymmetric

Secure communication and verifying authenticity

- **Securely send to one recipient**
 - Place item in box and lock with **public** key of the recipient
 - Only the recipient's private key can open it
- **Verify authenticity**
 - Place item in box and lock with private key of the owner
 - Magically copy box (its digital!)
 - Everyone can open the box using the owner's public key and know who put the item in the box

Certificates

- A Certificate says
 - Who we are
 - Our public key
 - When it's good until
 - What we are allowed to do with it
 - Who issued it and when

```
Data:
  Version: 3 (0x2)
  Serial Number:
02:37:17:3a:ae:17:19:16:88:3d:0a:0f:0e:b8:7d:e7
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C=US, O=Amazon, OU=Server CA 1B,
CN=Amazon
  Validity
    Not Before: Jul 10 00:00:00 2020 GMT
    Not After : Aug 10 12:00:00 2021 GMT
  Subject: CN=www.mongodb.com
  Subject Public Key Info:
    [... big hex number ... ]
    X509v3 Authority Key Identifier:

    X509v3 Subject Alternative Name:
      DNS:*.mongodb.com,
    X509v3 Key Usage: critical
      Digital Signature, Key Encipherment
    X509v3 Extended Key Usage:
      TLS Web Server Authentication
```

How can we trust a certificate?

It's signed

But how do I know the signature isn't fake?

Is it signed by the person who holds it (self-signed)

I need to know it was created by someone I trust.

What is a signature?

- Take the Certificate
- Compute a Hash of the data
 - A Message Digest – the MD in MD5
- Now Encrypt that MD with your private key
- Now anyone who has your public key knows you approved it.
- So a certificate comes with an encrypted MD
 - See who added the MD, get their public key
 - Decrypt, compare to actual MD, check they match

History of MongoDB security

- How to install MongoDB on AWS before 2017
 - `sudo yum install mongod`
 - That's it.
- Can anyone see the problem this causes?

Security in MongoDB

- Authentication
- Authorization
 - Views
- Transport Encryption
- Encryption-at-Rest
- Field-Level Encryption
- Auditing
- Log Redaction
- Network Restrictions

Authentication

Authentication is the act of proving you are who you say you are.

It normally is by proving one or more factors.

- Something you **Know** - like a password.
- Something you **Have** - like a certificate.
- Something you **Are** - like Biometrics, or an incoming IP address

Authentication Methods in MongoDB

- SCRAM-SHA
 - Salted Challenge Response Username and Password
- X.509
 - Certificates to prove identity
- LDAP
 - Authenticate Against Active Directory/OpenLDAP
- Kerberos
 - Authenticate with short-lived tokens/tickets
- MONGODB-AWS (Atlas Only)
 - Authenticate using Amazon Web Services IAM roles

SCRAM-SHA

- Username & salted password hash held in MongoDB
 - Client does not send the password to the server on login.
- Client requests a unique value (nonce) from server
 - Server Sends Back
 - Client hashes password, adds value hashes again
 - Client sends to Server
 - Server adds value to password hash and hashes
- This is One round of hashes on Client, Two on Server
 - Hash rounds are deliberately really slow and expensive – why?
- After login connection remains authenticated.

SCRAM-SHA – Pros and Cons

Pros

- It's simple.
- It's easy to understand.
- It's secure.

Cons

- We have to manage it separately for each database cluster.
- Authorization is local (added responsibility)
- What if the passwords become known by people?
- How to secure passwords for service users (i.e. the application)?

X.509 – Pros and Cons

Pros

- Can centrally provision credentials.
- We can have a revocation list (revoked certificates) or OCSP

Cons

- You need a separate mechanism for Authorization
 - This could be LDAP or Local
- To be secure, certificates also need to have a password.
 - So you still need a password store.

LDAP

- Federate out Authentication to Active Directory/LDAP
 - Username and Password verified by AD server.
 - Can enforce password complexity / expiry.
 - Database does rely on availability of Active Directory to enable login though.

LDAP and TLS

- LDAP uses plaintext credentials when verifying the user.
- It is strongly advised to configure LDAP to always use TLS to protect credentials on-the-wire.

LDAP – Pros and Cons

Pros

- Active Directory is frequently available in organizations.
- Allows users to log in with the same password they use for their desktop.
- Provides a single mechanism for Authentication and Authorization.
- Makes it easy to add or remove users from all databases.

Cons

- Relies on the Active Directory server being up.
 - Supports alternate servers.
- Configuration requires an understanding of your LDAP Tree
- Requires TLS connection to LDAP to be secure.

Kerberos Authentication

- Well regarded and proven.
- Kerberos is a bit like X.509
 - However new tickets (certs) are requested frequently
 - Users log in to a Kerberos Server to get a ticket that can be used to log into MongoDB**
 - Needs Kerberos Infrastructure set up to use it.

Kerberos – Pros and Cons

Pros

- Well respected security technology.
- Included in Active Directory
- Allows Windows users to login without a password
 - Passes their windows identity over.
 - This is especially good for Services as no need to store passwords.

Cons

- Requires a mechanism to get Authorization such as LDAP
- Requires users to be explicitly identified in the local instance.

What to Use and When

- Human users
 - Humans should always have individual logins
 - Centralized Identity Management (LDAP/Kerberos) is preferable.
 - SCRAM-SHA is secure but not centralized so requires more work to update credentials.
 - X.509 needs certificates with passwords and either local or LDAP Authorization.
- Software/Service users
 - No human should know the credentials used by a service.
 - Kerberos is a great option for Windows Services.
 - On Atlas with AWS we can use also IAM
 - Or generate a random password (and associated user) at install time
 - Store passwords securely at the application end - for example in an enterprise password management system.

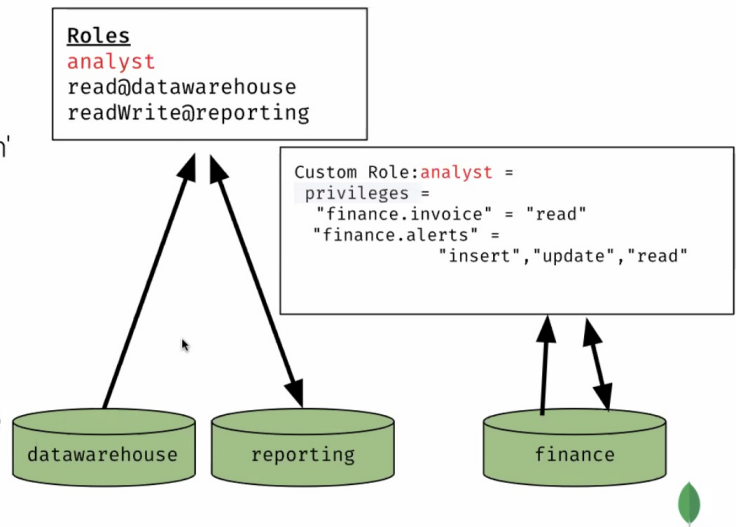
Authorization aka AUTHZ



sally@admin

What are you allowed to do?

- User is defined by username@database
 - Database *should* always be 'admin'
- Users have 1 or more Roles
 - Built-in roles or custom roles
 - Roles can 'inherit' from others.
 - Database roles or Atlas roles
- Roles include 1 or more privileges
 - A privilege means a user can perform an actions on a resource



right 2020-2021 MongoDB, Inc. All rights reserved.

Atlas user privileges

- Atlas admin
- Read and write to any database
- Only Read Any Database
- Custom Roles
 - Make more specific access for the user

Document Level Security Options

- MongoDB roles restrict down to Collection / View level.
- How can we have document level security?
 1. Application Level
 - Application uses a single database user
 - Application server determines what a user can see and do
 - Most common approach of having a database user per customer seems wrong.
 2. Using Views
 - We can create views which give a redacted view of the data
 - These can be difficult to index well depending on the definition.
 - Good choice for data-masking in API's or BI Users
 3. Using Realm
 - MongoDB Realm manages application users
 - And provides rules or function based, field level security.

Network Encryption

- MongoDB supports mandatory or optional TLS
 - requireTLS - Client Must use TLS
 - preferTLS - Clients 'May use TLS, Replication will use TLS if possible'
 - allowTLS - Client May use TLS
 - disabled - The Server does not support TLS
- MongoDB servers can optionally require client certificates.
 - If using Client Certificate these can optionally provide X.509 Authentication
 - Requiring valid server certificates when connection is good practice.
 - Requiring client certificates when not using X.509 Auth may be excessive in many cases.
- MongoDB advise **always** using Network Encryption when possible.
 - It's the default in Atlas and cannot be disabled.

Encryption at rest

- With Encryption at rest MongoDB Encrypts the files on disk.
- It needs a secure way to store and manage the decryption keys.
 - This means an external KMIP Key Vault
 - Storing Keys locally is not considered secure.
 - MongoDB will automatically rotate keys if using a Key Vault
- Encryption at rest protects us from the theft of files/disks
 - It does not protect us from anyone who has gained access to the machine.
 - It does not protect us from theft of files if they also take the key.
 - It is an alternative to using encrypted disk volumes if we don't have them available to us.