Securing Your Data: All steps for encrypting your MongoDB Database
About me

- Location: Skopje, Republic of Macedonia
- Education: MSc, Software Engineering
- Experience:
  - Lead Database Consultant (since 2016)
  - Database Consultant (2012 - 2016)
  - Web Developer, DBA (2007 - 2012)
- Certifications: C100DBA - MongoDB certified DBA (since 2016)

https://mk.linkedin.com/in/igorle  @igorle
Overview

• Security checklist
• Why encryption
• What are the prerequisites to set up encryption
• Step by step for transport encryption and at rest
• Encrypting data with volume encryption
• Percona for MongoDB feature comparison
• QA
Security incidents

- **Data breach** is a security incident in which sensitive, protected or confidential data is copied, transmitted, viewed, stolen or used by an individual unauthorized to do so.
- Ransomware attacks

![Annual number of data breaches and exposed records in the United States from 2005 to 2017 (in millions)](chart)

**Sources**
- Identity Theft Resource Center; CyberScout

**Additional Information:** United States, Identity Theft Resource Center; CyberScout; 2005 to 2017
Access hardening
Security checklist

- Network hardening
- Authentication
- Authorization (role based access control)
Network hardening

- Database servers exposed to the internet
- No Firewall, VPN or VPC
- net.bindIp: 0.0.0.0
Network hardening

- No need for DB server to be exposed to the internet
- Add Firewall rules, VPN or VPC
- `net.bindIp: IP-address`

*MongoDB 3.6 bind to localhost by default*
Authentication

1. Request a protected resource
   security.authentication: disabled

2. Return requested resource
1. Request a protected resource

   security.authentication: enabled

2. Ask for Authentication
Authorization

- Role Based Access Control (RBAC)
  - Role - grants privileges to perform the specified actions on resource
  - Privilege - consists of a specified resource and the actions permitted on the resource
  - Resource - database, collection, set of collections, or the cluster

- Built-In roles and User-Defined roles
1. Request a protected resource

2. Request username/password

3. Send username/password

4. Resource allowed
   - Yes
   - No

5. Return requested resource
   - Yes
   - No

4. Access denied
Encryption
Why Encryption?

- Attacker can access data by monitoring traffic between the application and the database or by reading files directly.
Why Encryption?

- Protect Personally Identifiable Information (PII)
  - PCI DSS for managing cardholder information
  - HIPAA standards for managing healthcare information
  - GDPR for the protection of EU citizen data privacy (May 2018)
  - FISMA to ensure the security of data in the federal government
  - FERPA to protect the privacy of student education records
  - Others
<table>
<thead>
<tr>
<th></th>
<th>USA vs EU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Privacy laws change with each administration.</td>
</tr>
<tr>
<td>2</td>
<td>Individuals have little ownership of their online data, which allows large businesses to monetize consumer behavior and habits.</td>
</tr>
<tr>
<td>3</td>
<td>Privacy laws are often a messy combination of public regulation, private self-regulation, and legislation which varies by state.</td>
</tr>
<tr>
<td>4</td>
<td>Enforcement of privacy laws is carried out by several different government organizations, e.g. Federal Communications Commission (FCC) and Health Insurance Portability and Accountability Act (HIPAA).</td>
</tr>
<tr>
<td>5</td>
<td>Numerous privacy organizations exist to provide a legal framework, which ensure digital privacy to Americans. E.g. American Civil Liberties Union (ACLU) and the Electronic Frontier Foundation (EFF).</td>
</tr>
<tr>
<td>6</td>
<td>Companies can keep data indefinitely, depending on their own Terms of Service.</td>
</tr>
<tr>
<td>1</td>
<td>Privacy laws have less turnover when administrations change because most EU member states aren't as polarized as the US.</td>
</tr>
<tr>
<td>2</td>
<td>EU laws respect &quot;private and family life&quot; and allow citizens to delete their data.</td>
</tr>
<tr>
<td>3</td>
<td>Privacy laws are generally more comprehensive and geared towards consumers.</td>
</tr>
<tr>
<td>4</td>
<td>Enforcement of privacy laws is carried out by one authority, equally for all 28 member states.</td>
</tr>
<tr>
<td>5</td>
<td>Due to the nature of EU rights, fewer privacy organizations exist but there are The European Digital Rights (EDRI) and The European Privacy Association (EPA).</td>
</tr>
<tr>
<td>6</td>
<td>EU citizens have the &quot;right to be forgotten,&quot; meaning that search results can be removed if they are irrelevant or inadequate.</td>
</tr>
</tbody>
</table>
Encryption with MongoDB

- Transport encryption
  - MongoDB network traffic is only readable by the intended client
- Encryption at REST
  - Application Level Encryption and Storage Encryption
  - Native encryption option for the WiredTiger storage engine*

* Available in MongoDB Enterprise only
Transport Encryption
Transport encryption

- TLS/SSL (Transport Layer Security/Secure Sockets Layer) to encrypt all of MongoDB’s network traffic
- Certificate Authorities - valid certificates generated and signed by a single certificate authority
  - PEMKeyfile with the name of the .pem file that contains the signed TLS/SSL certificate and key
  - CAFile with the name of the .pem file that contains the root certificate chain from the Certificate Authority
Transport encryption

Configuration notes: Deploying a 3 node replica set

MongoDB config file

# /etc/mongod.conf
systemLog:
  destination: file
  logAppend: true
  path: /mongodb/logs/mongod.log

storage:
  dbPath: /mongodb/data
  engine: wiredTiger
  wiredTiger:

net:
  port: 27017
  ssl:
    mode: <disabled|allowSSL|preferSSL|requireSSL>
    PEMKeyFile: /etc/ssl/mongodb.pem
    CAFFile: /etc/ssl/ca.pem

replication:
  replSetName: production
# Transport encryption

**net.ssl.mode**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabled</td>
<td>The server does not use TLS/SSL</td>
</tr>
<tr>
<td>allowSSL</td>
<td>Connections between servers do not use TLS/SSL. For incoming connections, the server accepts both TLS/SSL and non-TLS/non-SSL</td>
</tr>
<tr>
<td>preferSSL</td>
<td>Connections between servers use TLS/SSL. For incoming connections, the server accepts both TLS/SSL and non-TLS/non-SSL</td>
</tr>
<tr>
<td>requireSSL</td>
<td>The server uses and accepts only TLS/SSL encrypted connections</td>
</tr>
</tbody>
</table>
Transport encryption

1. Install MongoDB on each node*
2. Start each server with config file options for:
   `net.ssl.mode <allowSSL|preferSSL|requiredSSL>`
   `/usr/bin/mongod -f /etc/mongod.conf`
3. Initiate the replica set on the Primary node
4. Add the rest of the nodes by using FQDN

* confirm your MongoDB legacy supports TLS/SSL
Transport encryption

1. Install MongoDB on each node
2. Start each server with config file options for
   net.ssl.mode <allowSSL|preferSSL|requireSSL>
   /usr/bin/mongod -f /etc/mongod.conf
3. Initiate the replica set on the Primary node
4. Add the rest of the nodes by using FQDN
Transport encryption

1. Install MongoDB on each node
2. Start each server with config file options for
   net.ssl.mode <allowSSL|preferSSL|requireSSL>
   /usr/bin/mongod -f /etc/mongod.conf
3. Initiate the replica set on the Primary node
4. Add the rest of the nodes by using FQDN
Transport encryption

1. Install MongoDB on each node
2. Start each server with config file options for
   net.ssl.mode <allowSSL|preferSSL|requireSSL>
   /usr/bin/mongod -f /etc/mongod.conf
3. Initiate the replica set on the Primary node
4. Add the rest of the nodes by using FQDN
   rs.add( "blackbox.net:27017" )
Transport encryption

Upgrade running Replica to Use TLS/SSL

- Start the processes with ssl.Mode allowSSL
  
  ```
  net:
    ssl:
      mode: allowSSL
  ```

- Switch the clients to use TLS/SSL
Transport encryption

Upgrade running Cluster to Use TLS/SSL

- Start the processes with `ssl.Mode allowSSL`
  ```
  net:
  ssl:
    mode: allowSSL
  ```

- Switch the clients to use TLS/SSL

- Upgrade to `preferSSL` by issuing the command on each node
  ```
  db.adminCommand( { setParameter: 1, sslMode: "preferSSL" } )
  ```
Transport encryption

Upgrade running Cluster to Use TLS/SSL

- Start the processes with ssl.Mode allowSSL

  ```
  net:
    ssl:
      mode: allowSSL
  ```

- Switch the clients to use TLS/SSL

- Upgrade to preferSSL by issuing the command on each node

  ```
  db.adminCommand( { setParameter: 1, sslMode: "preferSSL" } )
  ```

- Upgrade to requireSSL by issuing the command on each node

  ```
  db.adminCommand( { setParameter: 1, sslMode: "requireSSL" } )
  ```

- Update the config file to persist the settings

  ```
  net.ssl.mode: requireSSL
  ```
Encryption at REST
Application level encryption

- Encryption on a per-field or per-document basis within the application layer
- Custom encryption and decryption routines to encrypt
  - Document
  - Field level data
Storage engine encryption

- Native encryption option for the WiredTiger storage engine only
- AES256-CBC (or 256-bit Advanced Encryption Standard in Cipher Block Chaining mode) via OpenSSL
- AES-256 uses a symmetric key; i.e. the same key to encrypt and decrypt text
Use of local key management via a keyfile

- Create the base64 encoded keyfile with the 16 or 32 character string

  `openssl rand -base64 32 > mongodb-keyfile`

- Assign permissions 600

  `chmod 600 mongodb-keyfile`

- Start mongod with the encryption options

  `/usr/bin/mongod --enableEncryption --encryptionKeyFile mongodb-keyfile`

  # /etc/mongod.conf

  security:
    enableEncryption: true
    encryptionKeyFile: /etc/mongodb_keyfile`
Encryption at REST (2)

- Integration with a third party key management appliance via the Key Management Interoperability Protocol (KMIP)
  - Key manager must support the KMIP communication protocol
  - Must have a valid certificate issued by the key management appliance

/usr/bin/mongod --enableEncryption --kmipServerName <KMIP Server HostName> --kmipPort <KMIP server port> --kmipServerCAFile ca.pem --kmipClientCertificateFile client.pem
Encryption with KMIP

- Generating a master key
- Generating keys for each database
- Encrypting data with the database keys
- Encrypting the database keys with the master key
- Master key and database keys are not replicated
Disk level encryption

- Amazon EBS encryption
- Azure disk encryption
- Google Compute Engine encrypts all data at rest (by default)
- Linux hard disk encryption with LUKS
- BitLocker encryption for Windows server
# Security features comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>MongoDB Community</th>
<th>MongoDB Enterprise</th>
<th>Percona server for MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP Authentication</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LDAP Authorization</td>
<td>✗</td>
<td>✓</td>
<td>✓ [1]</td>
</tr>
<tr>
<td>Kerberos</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Storage engine encryption</td>
<td>✗</td>
<td>✓</td>
<td>✓ [2]</td>
</tr>
<tr>
<td>Auditing</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Log redaction</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

[1] Requires additional script

[2] Uses native operating system disk encryption
Summary

• Security incidents and data breaches grow exponentially over the last 5 years
• Harden your database by limiting network exposure
• Enable access control by turning on authentication
• Limit resources access by authorization and roles
• Protect data in transit by using encryption with TLS/SSL
• Protect data at rest by using encrypted storage engine
Questions?
We’re Hiring!

https://www.pythian.com/careers/
Rate my session

**Schedule**
Timezone: Europe/Berlin +02:00

11:20

- **TAP THE SESSION**
- **Introducing gh-ost: triggerless, painless, trusted online schema migrations**
  11:20 - 12:10, Matterhorn 2

**Details**
**Introducing gh-ost: triggerless, painless, trusted online schema migrations**
- **Rate & Review**
- **TAP TO RATE & REVIEW**
- **DESCRIBE**
  - gh-ost - a tool which changes the paradigm of MySQL online schema changes, designed to overcome today's limitations and difficulties in online migrations.

**Rate & Review**
**Tap a star to rate**

Feedback (optional)

**SUBMIT**

© 2018 Pythian. Confidential