MongoDB Security Workshop

MongoDB Security Workshop

Release 3.4

MongoDB, Inc.

May 26, 2017

Contents

1 Security Workshop

Lab: Security Workshop **(page 2)** Securing a full deployment end to end

1.1 Lab: Security Workshop

Learning Objectives

Upon completing this workshop, attendees will be able to:

- Secure application communication with MongoDB
- Understand all security authentication and authorization options of MongoDB
- Encrypt MongoDB data at rest using encrypted storage engine
- Feel comfortable deploying and securely configuring MongoDB

Introduction

In this workshop, attendees will install and configure a secure replica set on servers running in AWS.

- We are going to secure the backend communications using TLS/SSL
- Enable authorization on the backend side
- Encrypt the storage layer
- Make sure that there are no *"leaks"* of information

Exercise: Accessing your instances from Windows

- Download and install Putty from **http://www.putty.org/**
- Start Putty with: **All Programs > PuTTY > PuTTY**
- In **Session**:
  - In the **Host Name** box, enter **centos@<publicIP>**
  - Under **Connection type**, select **SSH**
- In **Connection/SSH/Auth**,
  - Browse to the **AdvancedAdministrator.ppk** file
- Click **Open**
- Detailed info at: Connect to AWS with Putty[1]

[1] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html

Exercise: Accessing your instances from Linux or Mac

- get your .pem file and close the permissions on it

```
chmod 600 AdvancedAdministrator.pem
```

- enable the keychain and `ssh` into `node1`, propagating your credentials

```
ssh-add -K AdvancedAdministrator.pem
ssh -i AdvancedAdministrator.pem -A centos@54.235.1.1
```

- ssh into `node2` from `node1`

```
ssh -A node2
```

Solution: Accessing your instances

In our machines we will have access to all nodes in the deployment:

```
cat /etc/hosts
```

A `/share/downloads` folder with all necessary software downloaded

```
ls /share/downloads
ls /etc/ssl/mongodb
```

Exercise: Starting MongoDB and configuring the replica set

- `/share/downloads/mongodb_packages` contains MongoDB 3.2 and 3.4
- installation instructions are at:
  - https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-red-hat/
- configure the 3 nodes as a replica set named **SECURED**
- use `node1`, `node2` and `node3` for your host names
- you *MUST* use a config file[2]

[2] https://docs.mongodb.com/manual/reference/configuration-options/

Starting MongoDB and configuring the replica set (cont)

- installation

```
sudo yum install -y mongodb-enterprise-server-3.4.2-1.el7.x86_64.rpm
sudo vi /etc/mongod.conf
sudo service mongod start
```

- configure the 3 nodes as a replica set named **SECURED**, change **bindIp** to the **10.0.0.X** address, plus **127.0.0.1**

```
replication:
    replSetName: SECURED
net:
    bindIp: 10.0.0.101,127.0.0.1
```

- initiate the replica set

```
cfg = { _id: "SECURED", version: 1, members: [ {_id: 0, host: "node1:27017"}, {_
↪id: 1, host: "node2:27017"}, {_id:2, host: "node3:27017"} ] }
rs.initiate(cfg)
rs.status()
```

Exercise: Check the Connection to MongoDB

Let's try to connect to our running MongoDB cluster.

```
mongo --host SECURED/node1,node2,node3
```

Exercise: Launch the Client Application

It's time to connect our client application.

- install the application:

```
cd ~
tar xzvf /share/downloads/apps/security_lab.tgz
cd mongo-messenger
npm install
npm start
```

- Connect to the public ip of your `node4` instance, port 8080
  - `http://NODE4-public-IP:8080`

How is the client application connecting to the database?

- the connection string used by the application is in `message.js` and looks like this:

```
const url = "mongodb://node1:27017,node2:27017,node3:27017/
    security-lab?replicaSet=SECURED"
```

- this will work, for now...

WARNING: Spying your deployment!

Throughout the lab, the instructor will be spying on your deployment!

This checking is done by running a few scripts on your machines that will verify whether or not you have completely secured your deployment.

We will come back to this later on.

Exercise: Set up Authentication

Once we have our sample application up an running is time to start securing the system.

You should start by enabling MongoDB authentication[3]

To do this, you will have to decide:

- Which authentication mechanism to use
- Which authorization support will you use
- Set of users required to operate this system

Exercise: Enable SSL between the nodes

- we restricted "bindIp" to a local network interface, however if this was an outside address, it would not be good enough
- let's ensure we limit the connections to a list of nodes we control
  - let's use SSL certificates
  - they are in `/share/downloads/certs`
- http://mongodb.github.io/node-mongodb-native/2.2/tutorials/connect/ssl/

[3] https://docs.mongodb.com/manual/core/authentication/

Exercise: Encrypt Storage Layer

To fully secure our MongoDB deployment we need to consider the actual MongoDB instance files. Your instructor has some scripts that will enable him to have a peek into the your collection and indexes data files.

Don't let him do so!!!

Exercise: Avoid any log leaks

Logs are an important asset of your system.

Allow us to understand any potential issue with our cluster or deployment. But they can also **leak** some confidential information!

Make sure that you do not have any data leaks into your logs.

This should be done without downtime

Auditing

At this point we have a secured MongoDB deployment hardened against outside attacks, and used Role-Based Access Control to limit the access of users. The final step is to enable auditing, giving us a clear record of **who** performed an auditable action.

Exercise: Enable Auditing

- Enable auditing for all operations, to include CRUD operations, for your mongo-messenger user
- Output the log file in JSON format
- A log file has been created for you at `/mongod-data/audit/SECURED/audit.json`
- There are many filter options[4]

Putting it together

```
net:
  ssl:
    mode: requireSSL
    PEMKeyFile: /etc/ssl/mongodb/node1.pem
    CAFile: /etc/ssl/mongodb/ca.pem

security:
  clusterAuthMode: x509
  enableEncryption : true
  encryptionKeyFile : /etc/ssl/mongodb/mongodb-keyfile
  redactClientLogData: true

setParameter: { auditAuthorizationSuccess: true }

auditLog:
  destination: "file"
  format: "JSON"
```

[4] https://docs.mongodb.com/manual/tutorial/configure-audit-filters/

```
  path: /mongod-data/audit/SECURED/audit.json
  filter: '{ users: { user: "mongo-messenger", db: "security-lab" } }'
```

Summary

What we did:

- enabled basic authorization
- used SSL certificates
- encrypted the database at rest
- redacted the mongod logs
- configured auditing for a specific user

Find out more
mongodb.com | mongodb.org
university.mongodb.com

Having trouble?
File a JIRA ticket:
jira.mongodb.org

Follow us on twitter
@MongoDBInc
@MongoDB