Activer le chiffrement (SSL) pour les communications d'un cluster MongoDB

Sommaire

Introduction

Cette procédure permet d'activer le chiffrement des communications :

- entre les membres du cluster,
- entre les clients (*Shinken*) et le cluster.

Pour éviter une interruption de service, **les étapes doivent être suivies dans l'ordre.**

Dans les sections qui vont suivre, la valeur du paramètre **net.ssl.mode** de mongo (*https://www.mongodb.com/docs/v3.0/reference/configuration-options/#net.ssl.mode*) permet de définir son comportement vis-à-vis des connexions entrantes et des connexions sortantes :

Valeur de net.ssl.mode	Connexions entrantes acceptées	Connexions sortantes établies
disabled	non chiffrées	non chiffrées
allowSSL	non chiffrées et chiffrées	non chiffrées
preferSSL	non chiffrées et chiffrées	chiffrées
requireSSL	chiffrées	chiffrées

Générer les certificats SSL

On utilise la version d'OpenSSL livrée par Shinken pour créer les certificats.

Sur un des nœuds du cluster, on se place dans un dossier qui va stocker tous nos certificats :

```
mkdir /etc/shinken/certs/mongodb
cd /etc/shinken/certs/mongodb
```

Création des certificats de l'autorité de certification

Remplacer **Organization Inc.** par le nom de la société du client (*éventuellement*).

```
/opt/shinken/openssl/bin/openssl genrsa 2048 > ca-key.pem
/opt/shinken/openssl/bin/openssl req -new -x509 -nodes -days 365000 -key ca-key.pem -out ca-cert.pem -subj "
/C=FR/L=Paris/O=Organization Inc./OU=Shinken MongoDB CA/CN=Shinken MongoDB CA"
```

Création des certificats pour les noeuds du cluster

Les certificats émis pour les membres du cluster doivent répondre aux contraintes suivantes (*https://www.mongodb.com/docs/v3.0/tutorial/configure-x509-member-authentication/#x509-member-certificate*) :

- Ils doivent tous être émis par la même autorité de certification (*ex: le certificat créé dans la section précédente*).
- Parmi les attributs suivants des certificats, l'un d'entre eux doit forcément être non vide, et ils doivent avoir une valeur identique pour tous les certificats :
  - **O** (*Organization*),
  - **OU** (*Organizational Unit*),
  - **DC** (*Domain Components*).

Pour chaque nœud du cluster, exécuter la commande suivante :

- en remplaçant **server1** par le nom de chaque nœud tel que définis dans la configuration du cluster.
- en remplaçant **Organization Inc.** par la valeur utilisée pour le certificat de l'autorité de certification ci-dessus.

```
/opt/shinken/openssl/bin/openssl req -newkey rsa:2048 -nodes -days 365000 -keyout server1-key.pem -out
server1-req.pem -subj "/C=FR/L=Paris/O=Organization Inc./OU=Shinken MongoDB Cluster/CN=server1"
/opt/shinken/openssl/bin/openssl x509 -req -days 365000 -set_serial "0x`openssl rand -hex 8`" -in server1-
req.pem -out server1-cert.pem -CA ca-cert.pem -CAkey ca-key.pem
cat server1-key.pem server1-cert.pem > server1.pem
```

Création des certificats pour les clients (*serveurs Shinken utilisant mongos*)

Les certificats, émis pour les clients (daemons ou modules) qui vont interroger le cluster, doivent répondre aux contraintes suivantes (*https://www.mongodb.com/docs/v3.0/core/security-x.509/#std-label-client-x509-certificates-requirements*) :

- Ils doivent tous être émis par la même autorité de certification que celle du cluster (*ex: le certificat créé dans la section ci dessus*).
- Chaque client doit avoir son propre certificat.
- parmi les attributs suivants des certificats, l'un d'entre eux doit avoir une valeur différente de celles utilisées pour le cluster :
  - **O** (*Organization*),
  - **OU** (*Organizational Unit*),
  - **DC** (*Domain Components*).

- Pour chaque serveur Shinken qui va se connecter au cluster via mongos, exécuter la commande suivante :
  - En remplaçant **client1** par le nom du serveur Shinken, tel qu'il est connu par le serveur DNS,
  - En remplaçant **Organization Inc.** par la valeur utilisée pour le certificat de l'autorité de certification ci-dessus.

```
/opt/shinken/openssl/bin/openssl req -newkey rsa:2048 -nodes -days 365000 -keyout client1-key.pem -out
client1-req.pem -subj "/C=FR/L=Paris/O=Organization Inc./OU=Shinken MongoDB Client/CN=client1"
/opt/shinken/openssl/bin/openssl x509 -req -days 365000 -set_serial "0x`openssl rand -hex 8`" -in client1-
req.pem -out client1-cert.pem -CA ca-cert.pem -CAkey ca-key.pem
cat client1-key.pem client1-cert.pem > client1.pem
```

Déployer les certificats sur les noeuds du cluster

Remplacer **node2** et **node3** par les **noms** ou **IP** des deux autres membres du cluster :

```
rsync -avP --delete /etc/shinken/certs/mongodb/ node2:/etc/shinken/certs/mongodb/
rsync -avP --delete /etc/shinken/certs/mongodb/ node3:/etc/shinken/certs/mongodb/
```

Faites une copie de tous les certificats, pour avoir en plus une sauvegarde des certificats.

Déployer les certificats sur les clients (*serveurs Shinken utilisant mongos*)

Remplacer **client1** par le nom ou l'**IP** du serveur Shinken qui va se connecter au cluster :

```
rsync -avP /etc/shinken/certs/mongodb/ca-cert.pem /etc/shinken/certs/mongodb/client1.pem client1:/etc/shinken
/certs/mongodb/
```

Activer le chiffrement sur le cluster

Autoriser le chiffrement des connexions au cluster

Pour éviter une coupure de service, intervenir sur chaque nœud du cluster, l'un après l'autre. L'arrêt d'un seul nœud sera transparent pour la production.

Sur cette première étape, nous allons configurer mongod pour accepter les connexions entrantes chiffrées, mais il se connectera toujours aux autres nœuds en clair.

- Sur chaque nœud du cluster, éditer */etc/mongod.conf* pour ajouter cet extrait à la configuration, en remplaçant **NODEX** par le nom de chaque nœud :

/etc/mongod.conf

```
net:
  ssl:
    mode: allowSSL
    PEMKeyFile: /etc/shinken/certs/mongodb/NODEX.pem
    CAFile: /etc/shinken/certs/mongodb/ca-cert.pem
```

- puis appliquer le changement :

```
service mongod restart
```

Chaque nœud accepte alors les connexions entrantes en clair ou chiffrées, et il établit ses connexions sortantes en clair.

Activer le chiffrement des connexions entre les membres du cluster

Nous allons maintenant configurer mongod pour lui dire d'établir ses connexions vers les autres membres du cluster avec le chiffrement. Les connexions entrantes sont toujours acceptées en clair et en chiffré.

- Sur chaque nœud du cluster, lancer un shell mongo vers les mongod :

```
mongo --port 27018
```

- puis exécuter la commande suivante :

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

- pour rendre la modification permanente, remplacer **allowSSL** par **preferSSL** dans */etc/mongod.conf*, inutile de redémarrer **mongod**, la configuration étant déjà appliquée :

```
net:
  ssl:
    mode: preferSSL
```

Chaque nœud accepte alors les connexions entrantes en clair ou chiffrées, et établit des connexions sortantes chiffrées.

À ce stade, les connexions du cluster sont chiffrées.

Autoriser le chiffrement des connexions aux mongo-configsrv

Pour éviter une coupure de service, intervenir sur chaque nœud du cluster, l'un après l'autre. L'arrêt d'un seul nœud sera transparent pour la production.

Sur cette première étape, nous allons configurer mongod pour accepter les connexions chiffrées et en clair. Les mongos se connectant toujours en clair pour le moment, nous préparons le terrain pour pouvoir les passer en connexions chiffrées.

- Sur chaque nœud du cluster, éditer */etc/mongo-configsrv.conf* pour ajouter cet extrait à la configuration, en remplaçant **NODEX** par le nom de chaque nœud :

/etc/mongo-configsrv.conf

```
net:
  ssl:
    mode: preferSSL
    PEMKeyFile: /etc/shinken/certs/mongodb/NODEX.pem
    CAFile: /etc/shinken/certs/mongodb/ca-cert.pem
```

- puis appliquer le changement :

```
service mongo-configsrv restart
```

Les mogo-configsrv acceptent alors les connexions entrantes chiffrées ou en clair.

Activer le chiffrement depuis les mongos (*clients Shinken*)

Maintenant que tous les démons du cluster acceptent les connexions chiffrées, nous allons configurer mongos pour établir des connexions chiffrées.

- Sur chaque nœud du cluster et sur chaque serveur Shinken se connectant au cluster, éditer le fichier */etc/mongos.conf* pour ajouter cet extrait à la configuration, en remplaçant **NODEX** par le nom du serveur où est faite l'édition :

/etc/mongos.conf

```
net:
  ssl:
    mode: preferSSL
    PEMKeyFile: /etc/shinken/certs/mongodb/NODEX.pem
    CAFile: /etc/shinken/certs/mongodb/ca-cert.pem
```

- puis appliquer le changement :

```
service mongos restart
```

Les mongos acceptent des connexions entrantes chiffrées ou en clair, et ils établissent des connexions sortantes (*vers le cluster*) uniquement chiffrées.

Forcer le chiffrement des connexions aux mongo-configsrv

Une fois les mongos configurés pour utiliser le chiffrement, on peut bloquer les connexions entrantes en clair sur les mongo-configsrv.

Pour éviter une coupure de service, intervenir sur chaque noeud du cluster, l'un après l'autre. L'arrêt d'un seul noeud sera transparent pour la production.

- Sur chaque noeud du cluster, éditer */etc/mongo-configsrv.conf* pour remplacer **preferSSL** par **requireSSL**

/etc/mongo-configsrv.conf

```
net:
  ssl:
    mode: requireSSL
```

- puis appliquer le changement

```
service mongo-configsrv restart
```

Les mogo-configsrv n'acceptent alors que les connexions entrantes chiffrées, et ils établissent des connexions sortantes chiffrées.

Forcer le chiffrement des connexions au cluster

Dernière étape, nous allons configurer mongod sur les nœuds du cluster pour n'accepter que des connexions chiffrées.

- Sur chaque nœud du cluster, lancer un shell mongo :

```
mongo --port 27018
```

- puis exécuter la commande suivante :

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

- pour rendre la modification permanente, remplacer **preferSSL** par **requireSSL** dans */etc/mongod.conf*, inutile de redémarrer **mongod**, la configuration étant déjà appliquée

/etc/mongod.conf

```
net:
  ssl:
    mode: requireSSL
```

Chaque noeud n'accepte que les connexions entrantes chiffrées, et établit des connexions sortantes chiffrées.

Chiffrer les connexions de Shinken au cluster

Shinken se connectant à mongos via la boucle locale (*127.0.0.1*), et mongos chiffrant ses communications vers le cluster, il n'y a rien à faire.

Les connexions de Shinken à mongos peuvent rester en clair. Le chiffrement n'aurait aucune valeur ajoutée, en plus de nécessiter un supplément de ressources CPU.

Pour ne pas se retrouver avec un mongos qui relaie des requêtes provenant de partout, le mettre en écoute uniquement sur la boucle locale.

/etc/mongos.conf

```
net:
  bindIp: 127.0.0.1
```

Utilisation du client mongo en ligne de commande

Si un mongos est présent sur la machine locale, il suffit de se connecter normalement en clair au mongos.

Depuis une machine **clientX**, pour établir une connexion chiffrée vers un mongod sur **serveurY**, voici la ligne de commande à utiliser

```
mongo --ssl --sslCAFile /etc/shinken/certs/mongodb/ca-cert.pem --sslPEMKeyFile /etc/shinken/certs/mongodb
/clientX.pem --port 27018 --host serveurY
```

Pour les connexions locales, il ne faut **pas** utiliser **localhost** ou **127.0.0.1** comme paramètre de --host, mais le nom du serveur tel qu'il est défini dans le certificat.

Exemple depuis la machine **serveurY** :

```
mongo --ssl --sslCAFile /etc/shinken/certs/mongodb/ca-cert.pem --sslPEMKeyFile /etc/shinken/certs
/mongodb/serveurY.pem --port 27018 --host serveurY
```

Supervision du cluster MongoDB en SSL

Shinken recommande de superviser le cluster MongoDB créé lors de la mise en place de la haute disponibilité pour la base de données de Shinken :

- Il faut tous d'abord mettre en place la supervision du cluster MongoDB (*voir la page Haute disponibilité de la base MongoDB (mise en place d'un cluster)*) .
- Puis adapter la supervision du cluster à l'activation du chiffrement (*voir la page* Supervision d'un cluster MongoDB)