Sécurisation des connexions aux bases MongoDB

Sommaire

Contexte

Pour stocker ses données, **Shinken Entreprise** utilise le système de base de données **MongoDB**.

- Cette page montre comment sécuriser les données de la base.
- Selon l'infrastructure, il est conseillé des stratégies de protection adaptée.

Une grande partie de la sécurisation de la base repose sur la limitation de l'écoute des requêtes sur la boucle locale (*localhost*).
Si la base n'écoute que sur la boucle locale, son accès est automatiquement limité et le chiffrement de ces connexions n'est plus utile (*on ne chiffre pas les données qui transitent sur la boucle locale*).

Infrastructure avec un Mongo Standalone

Infrastructure mono-serveur

S'il n'y a qu'une seule machine pour l'installation de Shinken, la configuration par défaut de la base (*disponible dans le fichier /etc/mongod.conf*) lui permet seulement d'écouter les requêtes sur l'interface locale (*localhost*).

Extrait de /etc/mongod.conf

```
# Listen to local interface only. Comment out to listen on all interfaces.
bind_ip=127.0.0.1
```

Cela implique qu'aucun autre serveur ne pourra se connecter directement à la base MongoDB, seuls les démons de la machine pourrons interroger MongoDB sur la boucle locale (*localhost*).

C'est le moyen le plus simple de sécuriser l'accès à la base.

Infrastructure multi-serveur avec une base MongoDB commune

Dans le cas, où il y a plusieurs machines sur l'infrastructure, Shinken préconise de laisser la configuration par défaut de la base (*disponible dans le fichier /etc/mongod.conf*) qui limite l'écoute à l'interface locale (*localhost*).

Extrait de /etc/mongod.conf

```
# Listen to local interface only. Comment out to listen on all interfaces.
bind_ip=127.0.0.1
```

Pour que les démons, qui ne sont pas sur la machine qui héberge le **MongoDB**, puissent y accéder, il est recommandé d'utiliser des tunnels SSH.

- Ce système est intégré à Shinken et la mise en place des tunnels nécessite de simplement configurer les démons et modules qui se connectent à la base et de déployer la clé SSH de Shinken sur ces machines.

Cette approche à deux avantages :

- limite l'écoute de la base à l'interface locale,
- chiffre les connexions au serveur qui héberge la base MongoDB.

Il est possible d'utiliser les clés SSH crées lors de l'installation de Shinken pour faire ces tunnels (*voir la page* Création automatique et gestion de la clé SSH de l'utilisateur shinken).

Si pour une contrainte interne, vous avez besoin de chiffrer la connexion sur la boucle locale (NON RECOMMANDÉ)

Les étapes suivantes décrivent comment activer le chiffrement SSL pour les connexions entre Shinken et la base MongoDB.

Cette opération entraînera nécessairement une **indisponibilité** de Shinken le temps de redémarrer celui-ci ainsi que de la base MongoDB.

> Toutefois, comme ce chiffrement ne présente aucun réel gain de sécurité — les communications se faisant soit au sein d'une même machine, soit via un tunnel SSH — et qu'il entraîne une surcharge en ressources ainsi qu'une complexité accrue, **Shinken déconseille fortement l'activation du SSL** dans ce contexte.

Création des certificats pour MongoDB et pour Shinken

Les différents certificats doivent être émis pas la même autorité de certification.

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 du certificat de MongoDB

Exécuter les commandes suivantes :

- en remplaçant **SERVER** par le nom du serveur MongoDB tel que définis dans la configuration de Shinken (*IP, localhost, nom 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 -keyout SERVER-key.pem -out SERVER-req.pem -
subj "/C=FR/L=Paris/O=ORGANIZATION INC./OU=Shinken MongoDB Cluster/CN=SERVER"
/opt/shinken/openssl/bin/openssl x509 -req -days 365000 -set_serial "0x`openssl rand -hex 8`"-in SERVER-req.
pem -out SERVER-cert.pem -CA ca-cert.pem -CAkey ca-key.pem
cat SERVER-key.pem SERVER-cert.pem > SERVER.pem
```

Il faut ensuite déployer le certificat sur le serveur MongoDB en adaptant la valeur **SERVER** :

```
rsync -avP --delete /etc/shinken/certs/mongodb/  SERVER:/etc/shinken/certs/mongodb/
```

Création du certificat de Shinken

- Remplacer le champ *subjectAltName* par la liste des serveurs hébergeant au moins un des démons Shinken suivants : **Synchronizer, Broker ou Scheduler** (*dans le cas de la retention MongoDB*).
  - Pour ajouter une IP : "IP:IP_DU_SERVEUR"
  - Pour ajouter un nom DNS : "DNS:NOM_DU_SERVEUR"

```
/opt/shinken/openssl/bin/openssl req -newkey rsa:2048 -nodes -keyout shinken-key.pem -out shinken-req.pem -subj "/C=FR/L=Paris" -addext 'subjectAltName = DNS:shinken-node1, IP:shinken-node2'
/opt/shinken/openssl/bin/openssl x509 -req -days 365000 -set_serial "0x`openssl rand -hex 8`" -in shinken-req.pem -out shinken-cert.pem -CA ca-cert.pem -CAkey ca-key.pem

cat shinken-key.pem shinken-cert.pem > shinken.pem
```

Ensuite, il est nécessaire de déployer le certificat sur **toutes les machines** hébergeant au moins un des démons Shinken suivants : **Synchronizer, Broker ou Scheduler** (*dans le cas de la retention MongoDB*).

Dans la commande suivante, adapter la valeur **SHINKEN_SERVER** :

```
rsync -avP /etc/shinken/certs/mongodb/shinken.pem SHINKEN_SERVER:/etc/shinken/certs/mongodb/
```

Activer le chiffrement SSL dans Shinken

Pour activer le chiffrement SSL dans Shinken, il faut préciser les paramètres de connexion dans le paramètre de **l'uri** de Mongo.

- Tous les composants de Shinken qui se connectent à MongoDB doivent voir leur configuration modifiée sur le serveur de l'Arbiter.

- Voici la liste des éléments qui se connectent à MongoDB et dont la configuration doit être mise à jour :
  - Le démon Synchronizer : (*voir la page* Paramètres globaux (synchronizer.cfg)) ;
  - Le module event-manager-reader : (*voir la page* Module event-manager-reader) ;
  - Le module event-manager-writer : (*voir la page* *Module event-manager-writer*) ;
  - Le module Graphite-Perfdata : (*voir la page* Module Graphite-Perfdata) ;
  - Le module MongoDB : (*voir la page* Module MongoDB) ;
  - Le module MongodbRetention : (*voir la page* Module MongodbRetention (Rétention en base de données centralisée par royaume)) ;
  - Le module SLA du Broker : (*voir la page* Module SLA) ;
  - Le module SLA de la WebUI : (*voir la page* Module SLA (WebUI)) ;
  - Le module livedata-module-sla-provider (*voir la page* Module livedata-module-sla-provider) ;
  - Le collecteur de type discovery-import (*voir la page* Collecteur de type discovery-import (Scan NMAP)) ;
  - Le module report-builder--module-sla-reader (*voir la page* Module report-builder--module-sla-reader) ;

- Dans le cas de l'utilisation de l'outil tiers Grafana, il faut aussi modifier sur la ou les machines avec un carbon-cache le fichier de configuration **/opt/graphite/conf/mongodb.conf** (*voir la page* Grafana - v8.3.2) ;

Paramètres d'uri	Description
`tls`	Active SSL/TLS pour les communications avec le mongos. Valeurs possibles : • true • false
`tlsAllowInvalidCertificates`	Accepter le certificat SSL de mongos même s'il est invalide, par exemple expiré. Valeurs possibles : • true • false
`tlsAllowInvalidHostnames`	Accepter le certificat SSL de mongos même si le nom d'hôte du certificat ne correspond pas à celui du serveur. Valeurs possibles : • true • false

`tlsCAFile`	Chemin vers le fichier de l'autorité de certification (*CA*) utilisé pour vérifier le certificat SSL du mongos.
`tlsCertificateKeyFile`	Chemin vers le fichier contenant le certificat SSL de Shinken.
`tlsCertificateKeyFilePassword`	Mot de passe du certificat SSL de Shinken . **Avertissement** Le mot de passe est utilisé en paramètre d'URL. Si il contient des caractères interdits dans une URL, ils devront être échappés (*URL encodés*). **caractère interdit** : / ? # [ ] @ ! $ & ' ( ) * + , ; = % (espace) **remplacement** %3A %2F %3F %23 %5B %5D %40 %21 %24 %26 %27 %28 %29 %2A %2B %2C %3B %3D %25 %20 ou + Pour plus d'information https://developer.mozilla.org/fr/docs/Glossary/percent-encoding et rfc3986 Exemple: pour le mot de passe ***ch@nge_me*** il faudra mettre **ch%40nge_me**
`tlsCRLFile`	Chemin vers le fichier CRL (*liste de révocation*) des certificats SSL à rejeter.

Exemple d'uri qui active le chiffrement SSL :

```
mongodb_uri=mongodb://localhost:27017/?safe=false&tls=true&tlsCertificateKeyFile=/etc/shinken/certs/mongodb/shinken.pem&tlsCAFile=/etc/shinken/certs/mongodb/ca-cert.pem
```

Forcer le chiffrement des connexions aux mongod

- Sur le serveur avec la base MongoDB, éditer */etc/mongod.conf* pour ajouter le paragraphe en adaptant la valeur **SERVER** :

/etc/mongod.conf

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

Redémarrage des mongo et de Shinken pour prendre en compte les modifications

Il faut dans l'ordre :

- éteindre Shinken sur tous les serveurs :

```
service-shinken stop
```

- Redémarrer tous mongodb:

```
service mongod restart
```

- Redémarrer Shinken sur tous les serveurs : :

```
service-shinken start
```

Infrastructure avec un cluster MongoDB

Infrastructure multi-serveur avec un cluster MongoDB

Ce chapitre suppose d'être déjà familier avec les composants d'un cluster MongoDB (*voir la page Haute disponibilité de la base MongoDB (mise en place d'un cluster)*).

Protection entre Shinken et mongos

Connexion de Shinken au cluster sur la boucle local (127.0.0.1)

Vu qu'il faut un **mongos** sur chaque machine qui contient un démon ou un module de Shinken qui a besoin d'un accès au cluster MongoDB, il est possible de limiter l'écoute des mongos à l'interface locale (*localhost*).

Extrait de /etc/mongos.conf

```
# Listen to local interface only. Comment out to listen on all interfaces.
bind_ip=127.0.0.1
```

Dans ce cas, dans la configuration de Shinken, toutes les connexions à la base se feront sur l'adresse localhost.

Cette configuration va provoquer une erreur dans la validation de la configuration des modules de rétention.

Il faut autoriser "localhost" comme adresse valide avec l'option "mongodb_retention__database__bypass_banning_localhost_uri" de la configuration du module MongodbRetention (*voir la page Module MongodbRetention (Rétention en base de données centralisée par royaume)*).

Si pour une contrainte interne, vous avez besoin de chiffrer la connexion sur la boucle local (NON RECOMMANDÉ)

Ce chiffrement ne présente aucun réel gain de sécurité — les communications se faisant au sein d'une même machine — et entraîne une surcharge en ressources ainsi qu'une complexité accrue. **Shinken déconseille fortement l'activation du SSL** dans ce contexte (*voir la page* Activer le chiffrement (SSL) pour les communications d'un cluster MongoDB).

Protection entre mongos et les mongod/mongo-configsrv

Cette protection est assurée par :

- La mise en place des règles de firewall qui permettront de limiter l'accès à la base (*voir la page Haute disponibilité de la base MongoDB (mise en place d'un cluster)*).
- La mise en place du chiffrement des connexions (*voir la page Activer le chiffrement (SSL) pour les communications d'un cluster MongoDB*).

Activer l'authentification par mot de passe à la base de données MongoDB

Quelle que soit l'architecture, il est possible d'activer l'authentification par mot de passe à la base MongoDB :

- Dans le cas d'un seul serveur (*voir la page MongoDB - activation de l'authentification par mot de passe*).
- Dans le cas d'un cluster MongoDB (*voir la page Activer l'authentification par mot de passe dans un cluster MongoDB*).

Utilisation d'un antivirus

L'utilisation d'un antivirus peut avoir des conséquences néfastes sur le bon fonctionnement des démons MongoDB.

Se référer à la page Restrictions à appliquer aux antivirus pour configurer les exclusions à mettre en œuvre dans l'antivirus.