MongoDB security - Best practices

MongoDB

Mongo DB is included and used by BMC products. By default it has a very open configuration that is not secure for production environments. This document and steps included are designed to help the administrator set up and configure MongoDB to be secure so that the overall security posture of BMC products are also secure.

MongoDB security considerations when deploying for production use:

1. Deploy mongo as replica set

2a. Restrict Mongo DB access either by enabling authentication.

Or

2b. restricting the IP

- Explicitly bind mongod to a particular IP, such as a private network.
    - This is done via the bindIp configuration file option. (http://docs.mongodb.org/v2.4/reference/configuration-options/#bind_ip)
- Use a firewall or other network access control layer.
- Or a combination of the previous two items.

Note: You can consider doing both 2a and 2b too.

To Enable Auth for Mongo DB 2.4 Replica set

Note: You require a minimum of 3 mongo systems to setup auto fail over replica set for production usage.

Steps for mongodb set-up

1. Get root access

```
sudo su
```

On windows run as system administrator

2. Download Mongo DB 2.4.8 Release

```
curl -O http://downloads.mongodb.org/linux/mongodb-linux-x86_64-2.4.8.tgz
```

On windows - http://downloads.mongodb.org/win32/mongodb-win32-x86_64-2008plus-2.4.8.zip

3. Extract MongoDB From Archive

```
tar -zxvf mongodb-linux-x86_64-2.4.8.tgz
```

On windows unzip

4. Copy MongoDB to Target Directory

```
mkdir -p /dbdata/mongodb
cp -R -n mongodb-linux-x86_64-2.4.8/ /dbdata/mongodb
```

On windows move the unzipped folder to c:\dbdata\mongodb (create folder before moving)

5. Create db data directory

```
mkdir -p /dbdata/mongodata
```

On windows create folder c:\dbdata\mongodata

5b. create a key file

- The key file must be between 6 and 1024 characters and may only contain characters in the base64 set.
- The key file must not have group or "world" permissions on UNIX systems (chmod 600 <your-key-file>). On windows ignore this.
- Setting keyFile enables authentication and specifies a key file for the replica set members to use when authenticating to each other.
- The content of the key file is arbitrary but must be the same on all members of the replica set instances that connect to the set.

  Example File: /dbdata/mongodb/mongo.key

  AAAAB3NzaC1yc2EAAAABJQAAAQEAgeprnpVLAVu5n2lejNL0er5/ek/9fA5rse3z

  XGSgYSMfgBsl+UWHX6pgy/7ocOfwAurW4HRzv/EhzgqKoW0Tbv+Y5DGt7izekItQ

  h/vcGCmMr1YHl3ZogllcI7SCRqO0qnKKYi9OgNbtCmViQVmULYezY0uD+FZPmyGB

  TekQS0GhGUFN0umJMIRFGUKwSHNPK0g6hql955CojPQCcIErzVQrCdgT0Ow5Th0P

  AYibx9DAOgHQbhVlNYcVRiiVvfmZb0+PWTodu4DRNVHnklrkZCXJHoXmgrsC6jiS

  FdKkqbPdyGGZ8TOmtCa4P0DQVhv1DDSrDtVTHtjBOIQTAdUmWw

6a. Start mongodb server

```
/dbdata/mongodb/mongodb-linux-x86_64-2.4.8/bin/mongod --dbpath /dbdata/mongodata --port 27017
```

6b. connect to mongo shell and create admin user and social db admin

```
/dbdata/mongodb/mongodb-linux-x86_64-2.4.8/bin/mongo
> use admin
switched to db admin
> db.addUser( { user: "admin",pwd: "<your-password-goes-here>",roles: [ "userAdminAnyDatabase",
"clusterAdmin" , "readWriteAnyDatabase", "dbAdminAnyDatabase" ] } );
{
        "user" : "admin",
        "pwd" : "90f500568434c37b61c8c1ce05fdf3ae",
        "roles" : [
                "userAdminAnyDatabase",
                "clusterAdmin",
                "readWriteAnyDatabase",
                "dbAdminAnyDatabase"
        ],
        "_id" : ObjectId("54e2ed2097b8f594703fae17")
}
> use social
> db.createUser(
 { user: "social_admin", pwd: "<your-password-goes-here>", roles: [ "readWrite", "dbAdmin" ] }
)
```

Stop the mongo process that was started in Step 6a.

6c. Add MongoDB as service with key file parameter

Linux - RHEL-based distributions (including CENT OS)
Create a file **/etc/mongod.conf** and add the following

```
dbpath = /dbdata/mongodata
port = 27017
setParameter = textSearchEnabled=true
logpath = mongo_social.log
logappend = true
quiet = true
journal = true
replSet = myitsocial
fork = true
keyFile = /dbdata/mongodb/mongo.key
```

```
nohttpinterface = true
```

Create a file **/etc/init.d/mongorepl** with the appropriate permissions with following contents

```
#!/bin/sh
#
# /etc/init.d/mongorepl
# Subsystem file for "MyIT-MongoRepl" server
#
# chkconfig: 2345 95 05 (1)
# description: MyIT-MongoRepl server daemon
#MONGOBIN -> point to mongod binary.

MONGOBIN="/dbdata/mongodb/mongodb-linux-x86_64-2.4.8/bin/mongod"

do_start()
{
   echo "starting!";
   eval "$MONGOBIN -f /etc/mongod.conf";
}

do_stop()
{
   echo "stopping!"
   eval "$MONGOBIN -f /etc/mongod.conf --shutdown";
}

case "$1" in
   start)
     do_start
     ;;
   stop)
     do_stop
     ;;
esac

exit 0
```

And then use Redhat's chkconfig utility to automatically start the service

To add to chkconfig : chkconfig --add mongorepl

To enable in run levels : chkconfig mongorepl on

to start: service mongorepl start

to stop: service mongorepl stop

Start the mongo service

Windows:

Create a file c:\dbdata\mongodb\mongodb-win32-x86_64-2008plus-2.4.8\mongod.cfg add following content

```
dbpath=c:\dbdata\mongodata
port=27017
setParameter=textSearchEnabled=true
logpath=mongo_social.log
logappend=true
quiet=true
journal=true
replSet=myitsocial
keyFile = /dbdata/mongodb/mongo.key
nohttpinterface = true
```

Run cmd prompt as administrator

and issue following command

```
> c:\dbdata\mongodb\mongodb-win32-x86_64-2008plus-2.4.8\bin\mongod.exe --config
c:\dbdata\mongodb\mongodb-win32-x86_64-2008plus-2.4.8\mongod.cfg --install
```

To start the service

> net start MongoDB

Steps to setup replication

1. Login to MongoDB1 (assume MongoDB1 with IP 10.20.30.40) system and do the following:

MongoDB1 : install mongo DB (Follow the Step 1 to 6 from Steps for mongodb set-up)

Run this command to check if mongo is running or not

ps -aef|grep mongo

You should be able to see something similar to this

rlakkamr 35368 35347 0 06:10 pts/0   00:00:00 grep mongo
root   46274   1 0 Aug08 ?   00:13:32 /dbdata/mongodb/mongodb-linux-x86_64-2.4.8/bin/mongod --replSet myitsocial \--logpath .1.log. --dbpath /dbdata/mongodata \--port 27017 --keyFile /dbdata/mongodb/mongo.key \--setParameter textSearchEnabled=true
once verified

go to mongo shell

```
/dbdata/mongodb/mongodb-linux-x86_64-2.4.8/bin/mongo
```

Issue the command:

```
> use admin
> db.auth("admin", "<your-password-goes-here>");
1
```

and

```
> rs.initiate()
{
        "info2" : "no configuration explicitly specified -- making one",
        "me" : "10.20.30.40:27017",
        "info" : "Config now saved locally.  Should come online in about a minute.",
        "ok" : 1
}
>
```

wait for a minute

verify in same window by giving below command

rs.conf()

```
> rs.conf()
{
        "_id" : "social",
        "version" : 1,
        "members" : [
                {
                        "_id" : 0,
                        "host" : "10.20.30.40:27017"
                }
        ]
}
```

2. follow the same steps 1 - 6 (skip 6a and 6b) to other two systems
(Assume MongoDB2 with IP 10.20.30.41 and MongoDB3 with IP 10.20.30.42)

go to mongo shell MongoDB1

/dbdata/mongodb/mongodb-linux-x86_64-2.4.8/bin/mongo

Issue the command:

```
> db.auth("admin", "<your-password-goes-here>");
1
```

and to add member

```
>rs.add("10.20.30.41:27017")
>rs.add("10.20.30.42:27017")
```

To check the status of the replica setup issue the following command

```
rs.status()
```

You should be able to see the output something similar to this

```
{
"set" : "myitsocial",
"date" : ISODate("2014-08-12T06:15:02Z"),
"myState" : 1,
"members" : [
{
                        "_id" : 0,
                        "name" : "10.20.30.40:27017",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 303165,
                        "optime" : Timestamp(1407521114, 1),
                        "optimeDate" : ISODate("2014-08-08T18:05:14Z"),
                        "self" : true
                },
{
                        "_id" : 1,
                        "name" : "10.20.30.41:27017",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 302985,
                        "optime" : Timestamp(1407521114, 1),
                        "optimeDate" : ISODate("2014-08-08T18:05:14Z"),
                        "lastHeartbeat" : ISODate("2014-08-12T06:15:02Z"),
                        "lastHeartbeatRecv" : ISODate("2014-08-12T06:15:02Z"),
                        "pingMs" : 0,
                        "syncingTo" : "10.20.30.40:27017"
                },{
                        "_id" : 2,
                        "name" : "10.20.30.42:27017",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 302985,
                        "optime" : Timestamp(1407521114, 1),
                        "optimeDate" : ISODate("2014-08-08T18:05:14Z"),
                        "lastHeartbeat" : ISODate("2014-08-12T06:15:02Z"),
                        "lastHeartbeatRecv" : ISODate("2014-08-12T06:15:02Z"),
                        "pingMs" : 0,
                        "syncingTo" : "10.20.30.40:27017"
                }
],
"ok" : 1
}
```

3. (Optional to rotate mongodb logs)

Log rotate enable from admin schema

```
use admin
db.runCommand( { logRotate : 1} )
```

Make sure ulimit is set to 64000 in linux systems

```
ulimit -n 64000
```

Change the social config.js file in each MyIT/SmartIT server.

“db_host”: "mongodb://**social_admin**:<your-password-goes-here>@<mongo1 IP:27017>,<mongo2 IP:27017>,<mongo3 IP:27017>/",
"db_name": "**social**",

To Enable Auth for Mongo DB 2.4 standalone (if not running as replica)

1) Login in to mongo shell and follow the below instructions

```
use admin
```

```
db.addUser( { user: "admin",pwd: "password",roles: [ "userAdminAnyDatabase" ] } )
```

```
use social
```

```
db.addUser( { user: "social_admin",pwd: "password",roles: [ "readWrite", "dbAdmin" ]} )
```

2) Then enable mongodb authentication either start mongod process with `--auth` option or set `auth = true` in mongo config and restart mongo service

3) Change the config.js of Smart IT/MyIT social to connect to mongo

`"db_host":` “`mongodb://social_admin:password@<ip>:<port>/`”

Restart social service

Network Layer security

Configure Linux iptables Firewall for MongoDB

On contemporary Linux systems, the `iptables` program provides methods for managing the Linux Kernel's `netfilter` or network packet filtering capabilities. These firewall rules make it possible for administrators to control what hosts can connect to the system, and limit risk exposure by limiting the hosts that can connect to a system.

Rules in `iptables` configurations fall into chains, which describe the process for filtering and processing specific streams of traffic. Chains have an order, and packets must pass through earlier rules in a chain to reach later rules. This document addresses only the following two chains:

`INPUT`
: Controls all incoming traffic.

`OUTPUT`
: Controls all outgoing traffic.

Be aware that, by default, the default policy of `iptables` is to allow all connections and traffic unless explicitly disabled

Note: If you have configured different ports using the `port` configuration setting, you will need to modify the rules accordingly.

Traffic to and from `mongod` Instances

This pattern is applicable to all `mongod` instances running as standalone instances or as part of a replica set.

The goal of this pattern is to explicitly allow traffic to the `mongod` instance from the MyIT/Smart IT server. In the following examples, replace `<ip-address>` with the IP address of the MyIT/Smart IT server:

```
iptables -A INPUT -s <ip-address> -p tcp --destination-port 27017 -m state --state
NEW,ESTABLISHED -j ACCEPT


iptables -A OUTPUT -d <ip-address> -p tcp --source-port 27017 -m state --state ESTA
BLISHED -j ACCEPT
```

The first rule allows all incoming traffic from `<ip-address>` on port `27017`, which allows the application server to connect to the `mongod` instance. The second rule, allows outgoing traffic from the `mongod` to reach the application server.

Change Default Policy to `DROP`

The default policy for `iptables` chains is to allow all traffic. After completing all `iptables` configuration changes, you *must* change the default policy to `DROP` so that all traffic that isn't

explicitly allowed as above will not be able to reach components of the MongoDB deployment. Issue the following commands to change this policy:

```
iptables -P INPUT DROP


iptables -P OUTPUT DROP
```

Make all `iptables` Rules Persistent

By default all `iptables` rules are only stored in memory. When your system restarts, your firewall rules will revert to their defaults. When you have tested a rule set and have guaranteed that it effectively controls traffic you can use the following operations to you should make the rule set persistent.

On Red Hat Enterprise Linux, Fedora Linux, and related distributions you can issue the following command:

```
service iptables save
```

On Debian, Ubuntu, and related distributions, you can use the following command to dump the`iptables` rules to the `/etc/iptables.conf` file:

```
iptables-save > /etc/iptables.conf
```

Run the following operation to restore the network rules:

```
iptables-restore < /etc/iptables.conf
```

Place this command in your `rc.local` file, or in the `/etc/network/if-up.d/iptables` file with other similar operations.

Configure Windows netsh Firewall for MongoDB

On Windows Server systems, the `netsh` program provides methods for managing the **Windows Firewall**. These firewall rules make it possible for administrators to control what hosts can connect to the system, and limit risk exposure by limiting the hosts that can connect to a system.

Windows Firewall processes rules in an ordered determined by rule type, and parsed in the following order:

1. `Windows Service Hardening`
2. `Connection security rules`
3. `Authenticated Bypass Rules`
4. `Block Rules`
5. `Allow Rules`
6. `Default Rules`

By default, the policy in **Windows Firewall** allows all outbound connections and blocks all incoming connections.

Note: If you have configured different ports using the `port` configuration setting, you will need to modify the rules accordingly.

Traffic to and from `mongod.exe` Instances

This pattern is applicable to all `mongod.exe` instances running as standalone instances or as part of a replica set. The goal of this pattern is to explicitly allow traffic to the `mongod.exe` instance from the MyIT/Smart IT server.

```
netsh advfirewall firewall add rule name="Open mongod port 27017" dir=in action=allow protocol=TCP localport=27017
```

This rule allows all incoming traffic to port `27017`, which allows the MyIT/Smart IT server to connect to the `mongod.exe` instance.

Windows Firewall also allows enabling network access for an entire application rather than to a specific port, as in the following example:

```
netsh advfirewall firewall add rule name="Allowing mongod" dir=in action=allow program=" C:\mongodb\bin\mongod.exe"
```

Manage and Maintain *Windows Firewall* Configurations

This section contains a number of basic operations for managing and using `netsh`. While you can use the GUI front ends to manage the **Windows Firewall**, all core functionality is accessible is accessible from `netsh`.

Delete all *Windows Firewall* Rules

To delete the firewall rule allowing `mongod.exe` traffic:

```
netsh advfirewall firewall delete rule name="Open mongod port 27017" protocol=tcp l
ocalport=27017



netsh advfirewall firewall delete rule name="Open mongod shard port 27018" protocol
=tcp localport=27018
```

List All *Windows Firewall* Rules

To return a list of all **Windows Firewall** rules:

```
netsh advfirewall firewall show rule name=all
```

Reset *Windows Firewall*

To reset the **Windows Firewall** rules:

```
netsh advfirewall reset
```

Backup and Restore *Windows Firewall* Rules

To simplify administration of larger collection of systems, you can export or import firewall systems from different servers) rules very easily on Windows:

Export all firewall rules with the following command:

```
netsh advfirewall export "C:\temp\MongoDBfw.wfw"
```

Replace `"C:\temp\MongoDBfw.wfw"` with a path of your choosing. You can use a command in the following form to import a file created using this operation:

```
netsh advfirewall import "C:\temp\MongoDBfw.wfw"
```

To store MongoDB password in encrypted format in config file.

1. Run the following script

   \social\scripts\setmongodbpassword.js

   If you have already set the mongoDB username and password in the /social/config.js file
   “mongodb://social_admin:password@<ip>:<port>/”
   Please remove the username and password
   “mongodb://<ip>:<port>/”
   And then run the \social\scripts\setmongodbpassword.js script

   Usage:

   ```
   <node_path> setmongodbpassword.js <db_username> <db_password>
   ```

   After running the above script,
   \social\config.js file will be updated with the following attributes.
   "db_username": "social_admin", //user_name of DB
   "db_pw": "2738327c3d7eb81bc40b57c818dcbd62", // password will be encrypted using the auto generated cipher and will be handled in the application to decrypt.
   "db_auth": true // Will be set to true for authentication of mongoDB on the application.

2. After executing the script restart the social service