Center for Internet Security®

CIS MongoDB 3.2 Benchmark

v1.0.0 - 06-07-2017

Table of Contents

Overview

This document, CIS MongoDB Benchmark, provides prescriptive guidance for establishing a secure configuration posture for MongoDB version 3.2. This guide was tested against MongoDB 3.2 running on Ubuntu Linux 14.04, but applies to other linux distributions as well. To obtain the latest version of this guide, please visit http://benchmarks.cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write us at feedback@cisecurity.org.

Intended Audience

This document is intended for system and application administrators, security specialists, auditors, help desk, and platform deployment personnel who plan to develop, deploy, assess, or secure solutions that incorporate MongoDB.

Consensus Guidance

This benchmark was created using a consensus review process comprised of subject matter experts. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.
Each CIS benchmark undergoes two phases of consensus review. The first phase occurs during initial benchmark development. During this phase, subject matter experts convene to discuss, create, and test working drafts of the benchmark. This discussion occurs until consensus has been reached on benchmark recommendations. The second phase begins after the benchmark has been published. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in the benchmark. If you are interested in participating in the consensus process, please visit https://community.cisecurity.org.

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
Monospace font	Used for inline code, commands, or examples. Text should be interpreted exactly as presented.
<italic font in brackets>	Italic texts set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to denote the title of a book, article, or other publication.
Note	Additional information or caveats

Scoring Information

A scoring status indicates whether compliance with the given recommendation impacts the assessed target's benchmark score. The following scoring statuses are used in this benchmark:

Scored

Failure to comply with "Scored" recommendations will decrease the final benchmark score. Compliance with "Scored" recommendations will increase the final benchmark score.

Not Scored

Failure to comply with "Not Scored" recommendations will not decrease the final benchmark score. Compliance with "Not Scored" recommendations will not increase the final benchmark score.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1 - MongoDB on Linux**

  Items in this profile apply to MongoDB running on Linux and intend to:

  - be practical and prudent;
  - provide a clear security benefit; and
  - not inhibit the utility of the technology beyond acceptable means.

- **Level 2 - MongoDB on Linux**

  This profile extends the "Level 1 - MongoDB on Linux" profile. Items in this profile apply to MongoDB running on Linux and exhibit one or more of the following characteristics:

  - are intended for environments or use cases where security is paramount
  - acts as defense in depth measure
  - may negatively inhibit the utility or performance of the technology.

Acknowledgements

This benchmark exemplifies the great things a community of users, vendors, and subject matter experts can accomplish through consensus collaboration. The CIS community thanks the entire consensus team with special recognition to the following individuals who contributed greatly to the creation of this guide:

Author
Vinesh Redkar Security Consultant

Contributor
Chris Bielinski
Philippe Langlois, *Center for Internet Security*
Pralhad Chaskar

Editor
Karen Scarfone
Tim Harrison CISSP, ICP, *Center for Internet Security*

Recommendations

1 Installation and Patching

This section provides guidance on ensuring that the MongoDB software is up to date to eliminate easily avoidable vulnerabilities.

1.1 Ensure the appropriate MongoDB software version/patches are installed (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

The MongoDB installation version, along with the patch level, should be the most recent that is compatible with the organization's operational needs.

Rationale:

Using the most recent MongoDB software version along with all applicable patches helps limit the possibilities for vulnerabilities in the software. The installation version and/or patches applied should be selected according to the needs of the organization. At minimum, the software version should be supported.

Note: As of October 2016, only MongoDB versions 3.0 and 3.2 are still supported.

Audit:

Run the following command from within the MongoDB shell to determine if the MongoDB software version complies with your organization's operational needs:

```
> db.version()
```

Remediation:

Upgrade to the latest version of the MongoDB software:

1. Backup the data set.
2. Download the binaries for the latest MongoDB revision from the MongoDB Download Page and store the binaries in a temporary location. The binaries download as compressed files that extract to the directory structure used by the MongoDB installation.
3. Shutdown the MongoDB instance.
4. Replace the existing MongoDB binaries with the downloaded binaries.
5. Restart the MongoDB instance.

Default Value:

Patches are not installed by default.

References:

1. http://docs.mongodb.org/manual/tutorial/upgrade-revision/
2. https://docs.mongodb.com/manual/release-notes/
3. https://www.mongodb.com/download-center#community
4. https://www.mongodb.com/support-policy

CIS Controls:

4 – Continuous Vulnerability Assessment and Remediation

2 Authentication

This section contains recommendations for requiring authentication before allowing access to the MongoDB database.

2.1 Ensure that authentication is enabled for MongoDB databases (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

This setting ensures that all clients, users, and/or servers are required to authenticate prior to being granted access to the MongoDB database.

Rationale:

Failure to authenticate clients, users, and/or servers can enable unauthorized access to the MongoDB database and can prevent tracing actions back to their sources.

Audit:

Run the following command to verify whether authentication is enabled (`Auth` value set to `True`) on the MongoDB server.

```
cat /etc/mongod.conf | grep "Auth="
```

Remediation:

The authentication mechanism should be implemented before anyone accesses the MongoDB Server.

To enable the authentication mechanism:

- Start the MongoDB instance without authentication.

```
mongod --port 27017 --dbpath /data/db1]
```

- Create the system user administrator, ensuring that its password meets organizationally-defined password complexity requirements.

```
use admin
db.createUser(
{
    user: "siteUserAdmin",
    pwd: "password",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
  }
)
```

- Restart the MongoDB instance with authentication enabled.

```
mongod --auth --config /etc/mongod.conf
```

Default Value:

Not configured

References:

1. https://www.mongodb.com/blog/post/improved-password-based-authentication-mongodb-30-scram-explained-part-1
2. https://www.owasp.org/index.php/Authentication_Cheat_Sheet

CIS Controls:

16 – Account Monitoring and Control

2.2 Ensure that MongoDB does not bypass authentication via the localhost exception (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

MongoDB should not be set to bypass authentication via the localhost exception. The localhost exception allows you to enable authorization before creating the first user in the system.

Note: This recommendation only applies when there are no users created in the MongoDB instance.

Rationale:

Disabling this exception will prevent unauthorized local access to the MongoDB database. It will also ensure traceability of each database activity to a specific user.

Audit:

To verify the localhost exception is disabled, run the following command to ensure that `enableLocalhostAuthBypass` is set to `0` (`false`):

```
cat /etc/mongod.conf |grep "enableLocalhostAuthBypass"
```

Remediation:

Since `enableLocalhostAuthBypass` is not available using the `setParameter` database command, use the `setParameter` option in the configuration file to set it to `false`.

```
setParameter:
  enableLocalhostAuthBypass: false
```

Default Value:

Not configured

References:

1. http://docs.mongodb.org/manual/core/authentication/#localhost-exception

Notes:

The `--setParameter` option on the command line may also be used to configure this; however, having it in the config file may provide greater confidence that this setting is configured correctly in every instance.

CIS Controls:

16 – Account Monitoring and Control

2.3 Ensure authentication is enabled in the sharded cluster (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

Authentication is enabled in a sharded cluster when keyfiles are created and configured for all components. This ensures that every client that accesses the cluster must provide credentials, to include MongoDB instances that access each other within the cluster.

Rationale:

Enforcing a key on a sharded cluster prevents unauthorized access to the MongoDB database and provides traceability of database activities to a specific user or component.

Audit:

Run the following command to verify that the `keyfile` parameter is configured:

```
cat /etc/mongod.conf | grep "keyFile="
```

Remediation:

To enable authentication in the sharded cluster, perform the following steps:

- Generate a key file.
- On each component in the shared cluster, enable authentication by doing one of the following:
    - In the configuration file `/etc/mongod.conf`, set the `keyFile` option to the key file's path and then start the component with this command:

```
keyFile = /srv/mongodb/keyfile
```

- When starting the component, set `--keyFile` option, which is an option for both mongos instances and mongod instances. Set the `--keyFile` to the key file's path.

Default Value:

Not configured

References:

1. http://docs.mongodb.org/v2.2/administration/sharded-clusters/

CIS Controls:

16 – Account Monitoring and Control

2.4 Ensure an industry standard authentication mechanism is used (Scored)

Profile Applicability:

- Level 2 - MongoDB on Linux

Description:

Using one or more industry standard authentication mechanisms helps organizations enforce their account and password policies for their MongoDB users.

Rationale:

Without an industry standard authentication mechanism in place, account and password management is more tedious, and authentication may not align with the organization's policies.

Audit:

To verify the authentication mechanism in use for MongoDB, run the following commands:

```
cat /etc/mongod.conf | grep "clusterAuthMode:"
cat /etc/mongod.conf | grep "mode:"
cat /etc/mongod.conf | grep "authorization:"
cat /etc/mongod.conf | grep "authenticationMechanisms:"
```

Remediation:

In order to implement an industry standard authentication mechanism, use the corresponding sample from the list below as a model for specifying the authentication mechanisms in the MongoDB configuration file.

x.509 Certificates for Client Authentication:

```
security:
clusterAuthMode: x509
net:
ssl:
mode: requireSSL
PEMKeyFile: <path to TLS/SSL certificate and key PEM file>
CAFile: <path to root CA PEM file>
```

See the reference section for a link to a detailed procedure for generating the `PEMKeyFile` and `CAFile`.

MongoDB with Kerberos Authentication on Linux:

```
security:
   authorization: enabled
setParameter:
   authenticationMechanisms: GSSAPI
storage:
   dbPath: /opt/mongodb/data
```

See the reference section for a link to a detailed procedure for establishing the Kerberos service principal and `keytab` file.

References:

1. https://docs.mongodb.com/v3.2/tutorial/configure-x509-client-authentication/
2. https://docs.mongodb.com/v3.2/tutorial/control-access-to-mongodb-with-kerberos-authentication/
3. https://docs.mongodb.com/v3.2/core/kerberos/#kerberos-service-principal
4. https://docs.mongodb.com/v3.2/core/kerberos/#keytab-files

Notes:

Configuring the X.509 certificate and deploying Kerberos is beyond the scope of the document.

CIS Controls:

16 – Account Monitoring and Control

3 Access Control

This section contains recommendations for restricting access to MongoDB systems.

3.1 Ensure that role-based access control is enabled and configured appropriately (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

Role-based access control (RBAC) is a method of regulating access to resources based on the roles of individual users within an enterprise. A user is granted one or more roles that determine the user's access to database resources and operations. Outside of role assignments, the user has no access to the system. MongoDB can use RBAC to govern access to MongoDB systems. MongoDB does not enable authorization by default.

Rationale:

When properly implemented, RBAC enables users to carry out a wide range of authorized tasks by dynamically regulating their actions according to flexible functions. This allows an organization to control employees' access to all database tables through RBAC.

Audit:

Connect to MongoDB with the appropriate privileges and run the following command:

```
mongo --port 27017 -u <siteUserAdmin> -p <password> --authenticationDatabase
<database name>
```

Identify users' roles and privileges:

```
> db.getUser()
> db.getRole()
```

Verify that the appropriate role or roles have been configured for each user.

Remediation:

1. Establish roles for MongoDB.
2. Assign the appropriate privileges to each role.
3. Assign the appropriate users to each role.
4. Remove any individual privileges assigned to users that are now addressed by the roles.
5. See the reference below for more Information.

References:

1. http://docs.mongodb.org/manual/tutorial/manage-users-and-roles/

CIS Controls:

14.4 – Protect Information with Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.2 Ensure that MongoDB only listens for network connections on authorized interfaces (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

Ensuring that MongoDB runs in a trusted network environment involves limiting the network interfaces on which MongoDB instances listen for incoming connections. Any untrusted network connections should be dropped by MongoDB.

Rationale:

This configuration blocks connections from untrusted networks, leaving only systems on authorized and trusted networks able to attempt to connect to the MongoDB. If not configured, this may lead to unauthorized connections from untrusted networks to MongoDB.

Audit:

1. Verify that network exposure is limited, review the settings in the MongoDB configuration file:

```
cat /etc/mongod.conf |grep -A12 "net" | grep "bindIp"
```

2. Verify the relevant network settings on the Linux system itself:

```
iptables -L
```

Remediation:

Configure the MongoDB configuration file to limit its exposure to only the network interfaces on which MongoDB instances should listen for incoming connections.

Default Value:

Not configured

References:

1. http://docs.mongodb.org/manual/tutorial/configure-linux-iptables-firewall/
2. http://docs.mongodb.org/manual/tutorial/configure-windows-netsh-firewall/

CIS Controls:

9.1 – Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

3.3 Ensure that MongoDB is run using a non-privileged, dedicated service account (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

The MongoDB service should not be run using a privileged account such as 'root' because this unnecessarily exposes the operating system to high risk.

Rationale:

Using a non-privileged, dedicated service account restricts the database from accessing the critical areas of the operating system which are not required by the MongoDB. This will also mitigate the potential for unauthorized access via a compromised, privileged account on the operating system.

Audit:

Run the following command to get listing of all mongo instances, the PID number, and the PID owner.

```
ps -ef | grep -E "mongos|mongod"
```

Remediation:

1. Create a dedicated user for performing MongoDB database activity.
2. Set the Database data files, the keyfile, and the SSL private key files to only be readable by the mongod/mongos user.
3. Set the log files to only be writable by the mongod/mongos user and readable only by root.

Default Value:

Not configured

References:

1. http://docs.mongodb.org/manual/tutorial/manage-users-and-roles/

CIS Controls:

5.1 – Minimize and Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

3.4 Ensure that each role for each MongoDB database is needed and grants only the necessary privileges (Scored)

Profile Applicability:

- Level 2 - MongoDB on Linux

Description:

Reviewing all roles periodically and eliminating unneeded roles as well as unneeded privileges from necessary roles helps minimize the privileges that each user has.

Rationale:

Although role-based access control (RBAC) has many advantages for regulating access to resources, over time some roles may no longer be needed, and some roles may grant privileges that are no longer needed.

Audit:

Perform the following command to view all roles on the database on which the command runs, including both built-in and user-defined roles, as well as the privileges granted by each role. Ensure that only necessary roles are listed and only the necessary privileges are listed for each role.

```
db.runCommand(
    {
      rolesInfo: 1,
      showPrivileges: true,
      showBuiltinRoles: true
    }
)
```

Remediation:

To revoke specified privileges from the user-defined role on the database where the command is run. The `revokePrivilegesFromRole` command has the following syntax:

```
{
  revokePrivilegesFromRole: "<role>",
  privileges:
      [
        { resource: { <resource> }, actions: [ "<action>", ... ] },
        ...
      ],
}
```

References:

1. https://docs.mongodb.com/v3.2/reference/method/db.revokePrivilegesFromRole/
2. https://docs.mongodb.com/v3.2/reference/command/revokePrivilegesFromRole/#dbcmd.revokePrivilegesFromRole

Notes:

You must have the `dropRole` action on a database to drop a role from that database.

CIS Controls:

14.4 – Protect Information with Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.5 Review User-Defined Roles (Scored)

Profile Applicability:

- Level 2 - MongoDB on Linux

Description:

Reviewing all roles periodically and removing all users from those roles who do not need to belong to them helps minimize the privileges that each user has.

Rationale:

Although role-based access control (RBAC) has many advantages for regulating access to resources, over time some users may be assigned to roles that are no longer necessary, such as a user changing jobs within the organization. Users who have excessive privileges pose unnecessary risk to the organization.

Audit:

Check each role for each database using one of the following commands.

To specify a role from the current database, specify the role by its name:

```
db.runCommand( { rolesInfo: "<rolename>" } )
```

To specify a role from another database, specify the role by a document that specifies the role and database:

```
db.runCommand( { rolesInfo: { role: "<rolename>", db: "<database>" } } )
```

Remediation:

To remove a user from one or more roles on the current database, use the following command:

```
use <dbName>
db.revokeRolesFromUser( "<username>", [ <roles> ])
```

References:

1. https://docs.mongodb.com/manual/reference/method/db.revokeRolesFromUser/
2. https://docs.mongodb.com/manual/reference/command/rolesInfo/
3. https://docs.mongodb.com/manual/reference/privilege-actions/#authr.revokeRole

Notes:

Logged-in user must have the `revokeRole` action on a database to revoke a role on that database. Also, `roleInfo` works for both user-defined roles and built-in roles.

CIS Controls:

16.1 – Perform Regular Account Reviews

Review all system accounts and disable any account that cannot be associated with a business process and owner.

3.6 Review Superuser/Admin Roles (Scored)

Profile Applicability:

- Level 2 - MongoDB on Linux

Description:

Roles provide several advantages that make it easier to manage privileges in a database system. Security administrators can control access to their databases in a way that mirrors the structure of their organizations (they can create roles in the database that map directly to the job functions in their organizations). The assignment of privileges is simplified. Instead of granting the same set of privileges to each individual user in a particular job function, the administrator can grant this set of privileges to a role representing that job function and then grant that role to each user in that job function.

Rationale:

Reviewing the Superuser/Admin roles within a database helps minimize the possibility of privileged unwanted access.

Audit:

Superuser roles provide the ability to assign any user any privilege on any database, which means that users with one of these roles can assign themselves any privilege on any database:

```
db.runCommand( { rolesInfo: "dbOwner" } )
db.runCommand( { rolesInfo: "userAdmin" } )
db.runCommand( { rolesInfo: "userAdminAnyDatabase" } )
```

Root role provides access to the operations and all the resources of the `readWriteAnyDatabase`, `dbAdminAnyDatabase`, `userAdminAnyDatabase`, `clusterAdmin` roles, restore combined.

```
db.runCommand( { rolesInfo: "readWriteAnyDatabase" } )
db.runCommand( { rolesInfo: "dbAdminAnyDatabase" } )
db.runCommand( { rolesInfo: "userAdminAnyDatabase" } )
db.runCommand( { rolesInfo: "clusterAdmin" } )
```

Cluster Administration Roles are used for administering the whole system rather than just a single database.

```
db.runCommand( { rolesInfo: "hostManager" } )
```

Remediation:

To remove a user from one or more roles on the current database.

```
use <dbName>
db.revokeRolesFromUser( "<username>", [ <roles> ])
```

References:

1. https://docs.mongodb.com/v3.0/reference/built-in-roles/#built-in-roles
2. https://docs.mongodb.com/manual/reference/method/db.revokeRolesFromUser/

CIS Controls:

5.1 – Minimize and Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

16.1 – Perform Regular Account Reviews

Review all system accounts and disable any account that cannot be associated with a business process and owner.

4 Data Encryption

This section contains recommendations for securing data at rest (stored) and data in motion (transiting) for MongoDB.

4.1 Ensure TLS or SSL protects all network communications (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

Use TLS or SSL to protect all incoming and outgoing connections. This should include using TLS or SSL to encrypt communication between mongod and mongos components of a MongoDB client as well as between all applications and MongoDB.

Most MongoDB distributions include support for SSL or TLS.

Rationale:

This prevents sniffing of cleartext traffic between MongoDB components or performing a man-in-the-middle attack for MongoDB.

Audit:

To verify that the server requires SSL or TLS use (`net.ssl.mode` value set to `requireSSL`), run one of the following commands:

```
cat /etc/mongos.conf | grep –A20 'net' | grep –A10 'ssl' | grep 'mode'
```

Remediation:

Configure MongoDB servers to require the use of SSL or TLS to encrypt all MongoDB network communications.

To implement SSL or TLS to encrypt all MongoDB network communication, perform the following steps:

For mongod ("Primary daemon process for the MongoDB system")

In the configuration file /etc/mongod.conf, set the PEMKeyFile option to the certificate file's path and then start the component with this command:

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

And restart monogdb instance with

```
mongod --config /etc/mongod.conf
```

Or

```
mongod --sslMode requireSSL --sslPEMKeyFile /etc/ssl/mongodb.pem --sslCAFile
/etc/ssl/ca.pem
```

Default Value:

Not configured

References:

1. http://docs.mongodb.org/manual/tutorial/configure-ssl/

Notes:

Value	Description
disabled	The server does not use TLS/SSL.
allowSSL	Connections between servers do not use TLS/SSL. For incoming connections, the server accepts both TLS/SSL and non-TLS/non-SSL.
preferSSL	Connections between servers use TLS/SSL. For incoming connections, the server accepts both TLS/SSL and non-TLS/non-SSL.
requireSSL	The server uses and accepts only TLS/SSL encrypted connections.

CIS Controls:

14.2 – Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

4.2 Ensure Federal Information Processing Standard (FIPS) is enabled (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

The Federal Information Processing Standard (FIPS) is a computer security standard used to certify software modules and libraries that encrypt and decrypt data securely. You can configure MongoDB to run with a FIPS 140-2 certified library for OpenSSL.

Rationale:

FIPS is industry standard that dictates how data should be encrypted in rest and during transmission.

Audit:

To verify that the server uses FIPS Mode (`net.ssl.FIPSMode` value set to `true`), run following commands:

```
mongos --config /etc/mongos.conf
```

```
net:
 ssl:
 FIPSMode: true
```

or

To verify FIPS mode is running, check the server log file for a message that FIPS is active:

```
FIPS 140-2 mode activated
```

Remediation:

Configuring FIPS mode, ensure that your certificate is FIPS compliant. Run mongod or mongos instance in FIPS mode.

Make changes to configuration file, to configure your mongod or mongos instance to use FIPS mode, shut down the instance and update the configuration file with the following setting:

```
net:
   ssl:
      FIPSMode: true
```

Start mongod or mongos instance with a configuration file.

```
mongod --config /etc/mongod.conf
```

Default Value:

Not configured

References:

1. https://docs.mongodb.com/v3.2/tutorial/configure-fips/

CIS Controls:

14.2 – Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

14.5 – Encrypt at Rest Sensitive Information

Sensitive information stored on systems shall be encrypted at rest and require a secondary authentication mechanism, not integrated into the operating system, in order to access the information.

5 Auditing

This section contains recommendations related to configuring audit logging in MongoDB.

5.1 Ensure that system activity is audited (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

Track access and changes to database configurations and data. MongoDB Enterprise includes a system auditing facility that can record system events (e.g. user operations, connection events) on a MongoDB instance. These audit records permit forensic analysis and allow administrators to verify proper controls.

Rationale:

System level logs can be handy while troubleshooting an operational problem or handling a security incident.

Audit:

To verify that system activity is being audited for MongoDB, run the following command to confirm the `auditLog.destination` value is set correctly:

```
cat /etc/mongod.conf |grep –A4 "auditLog" | grep "destination"
```

Remediation:

Set the value of `auditLog.destination` to the appropriate value from the following options:

syslog

To enable auditing and print audit events to syslog

```
mongod --dbpath data/db --auditDestination syslog
```

console

To enable auditing and print audit events to standard output (i.e., `stdout`)

```
mongod --dbpath data/db --auditDestination console
```

Json File

To enable auditing and print audit events to a file in JSON format. Printing audit events to file in JSON format degrades server performance more than printing to a file in BSON format.

```
mongod --dbpath data/db --auditDestination file --auditFormat JSON --
auditPath data/db/auditLog.json
```

Bson File

To enable auditing and print audit events to a file in BSON binary format

```
mongod --dbpath data/db --auditDestination file --auditFormat BSON --
auditPath data/db/auditLog.bson
```

Default Value:

Not configured

References:

1. http://docs.mongodb.org/manual/tutorial/configure-auditing/

CIS Controls:

6.2 – Ensure Audit Log Settings Support Appropriate Log Entry Formatting

Validate audit log settings for each hardware device and the software installed on it, ensuring that logs include a date, timestamp, source addresses, destination addresses, and various other useful elements of each packet and/or transaction. Systems should record logs in a standardized format such as syslog entries or those outlined by the Common Event Expression initiative. If systems cannot generate logs in a standardized format, log normalization tools can be deployed to convert logs into such a format.

5.2 Ensure that audit filters are configured properly (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

MongoDB Enterprise supports auditing of various operations. When enabled, the audit facility, by default, records all auditable operations as detailed in Audit Event Actions, Details, and Results. To specify which events to record, the audit feature includes the `--auditFilter` option. This check is only for Enterprise editions.

Rationale:

All operations carried out on the database are logged. This helps in backtracking and tracing any incident that occurs.

Audit:

To verify that audit filters are configured on MongoDB as per the organization's requirements, run the following command:

```
cat /etc/mongod.conf |grep -A10 "auditLog" | grep "filter"
```

Remediation:

Set the audit filters based on the organization's requirements.

Default Value:

Not configured

References:

1. https://docs.mongodb.com/manual/reference/audit-message/
2. https://docs.mongodb.com/manual/tutorial/configure-audit-filters/

CIS Controls:

6.2 – Ensure Audit Log Settings Support Appropriate Log Entry Formatting

Validate audit log settings for each hardware device and the software installed on it, ensuring that logs include a date, timestamp, source addresses, destination addresses, and various other useful elements of each packet and/or transaction. Systems should record logs in a standardized format such as syslog entries or those outlined by the Common Event Expression initiative. If systems cannot generate logs in a standardized format, log normalization tools can be deployed to convert logs into such a format.

5.3 Ensure that logging captures as much information as possible (Not Scored)

Profile Applicability:

- Level 2 - MongoDB on Linux

Description:

The `SystemLog.quiet` option stops logging of information such as:

- connection events
- authentication events
- replication sync activities
- evidence of some potentially impactful commands being run (eg: `drop`, `dropIndexes`, `validate`)

This information should be logged whenever possible. This check is only for Enterprise editions.

Rationale:

The use of `SystemLog.quiet` makes troubleshooting problems and investigating possible security incidents much more difficult.

Audit:

To verify that the `SystemLog.quiet` option is disabled (value of False), run the following command:

```
cat /etc/mongod.conf |grep "SystemLog.quiet"
```

Remediation:

Set `SystemLog.quiet` to `False` in the `/etc/mongod.conf` file to disable it.

References:

1. https://docs.mongodb.com/manual/reference/configuration-options/#systemLog.quiet

CIS Controls:

6.2 – Ensure Audit Log Settings Support Appropriate Log Entry Formatting

Validate audit log settings for each hardware device and the software installed on it, ensuring that logs include a date, timestamp, source addresses, destination addresses, and various other useful elements of each packet and/or transaction. Systems should record logs in a standardized format such as syslog entries or those outlined by the Common Event Expression initiative. If systems cannot generate logs in a standardized format, log normalization tools can be deployed to convert logs into such a format.

5.4 Ensure that new entries are appended to the end of the log file (Not Scored)

Profile Applicability:

- Level 2 - MongoDB on Linux

Description:

By default, new log entries will overwrite old entries after a restart of the mongod or Mongols service. Enabling the `systemLog.logAppend` setting causes new entries to be appended to the end of the log file rather than overwriting the existing content of the log when the mongos or mongod instance restarts.

Rationale:

Allowing old entries to be overwritten by new entries instead of appending new entries to the end of the log may destroy old log data that is needed for a variety of purposes.

Audit:

To verify that new log entries will be appended to the end of the log file after a restart (`systemLog.logAppend` value set to `true`), run the following command:

```
cat /etc/mongod.conf |grep "systemLog.logAppend"
```

Remediation:

Set `systemLog.logAppend` to `true` in the `/etc/mongod.conf` file.

References:

1. https://docs.mongodb.com/manual/reference/configuration-options/#systemLog.logAppend

CIS Controls:

6.3 – Ensure Audit Logging Systems Are Not Subject to Loss (i.e. rotation/archive)
Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

6 Operating System Hardening

This section contains recommendations related to hardening the operating system running below MongoDB.

6.1 Mongodb Database Running with Least Privileges (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

This setting ensures that monogd service run as least privilege user.

Rationale:

Anyone who has been a victim of viruses, worms, and other malicious software (malware) will appreciate the security principle of "least privilege." If all processes ran with the smallest set of privileges needed to perform the user's tasks, it would be more difficult for malicious and annoying software to infect a machine and propagate to other machines.

Audit:

Connect MongoDB Service

```
mongod --port 27017 --dbpath /data/db1
```

It will highlight if the service is running as root privilege or not.

Remediation:

Create a user which is only used for running Mongodb and directly related processes. This user must not have administrative rights to the system.

Steps to create user

```
useradd -m -d /home/mongodb -s /bin/bash -g mongodb -u 1234 mongodb
```

And then set ownership to mongodb user only

```
sudo chown -R mongodb:mongodb /data/db
```

CIS Controls:

5.1 – Minimize and Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

6.2 Ensure that MongoDB uses a non-default port (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

Changing the port used by MongoDB makes it harder for attackers to find the database and target it.

Rationale:

Standard ports are used in automated attacks and by attackers to verify which applications are running on a server.

Audit:

To verify the port number used by MongoDB, execute the following command and ensure that the port number is not `27017`:

```
cat /etc/mongod.conf |grep "port"
```

Remediation:

Change the port for MongoDB server to a number other than `27017`.

Impact:

Hackers frequently scan IP addresses for commonly used ports, so it's not uncommon to use a different port to "fly under the radar". This is just to avoid detection, other than that there is no added safety by using a different port.

References:

1. https://docs.mongodb.com/manual/reference/default-mongodb-port/

CIS Controls:

9 – Limitation and Control of Network Ports, Protocols, and Services

6.3 Ensure that operating system resource limits are set for MongoDB (Not Scored)

Profile Applicability:

- Level 2 - MongoDB on Linux

Description:

Operating systems provide ways to limit and control the usage of system resources such as threads, files, and network connections on a per-process and per-user basis

Rationale:

These `ulimits` prevent a single user from consuming too many system resources.

Audit:

To verify the resource limits set for MongoDB, run the following commands.

Extract the process ID for MongoDB:

```
ps -ef|grep mongod
```

View the limits associated with that process number:

```
cat /proc/1322/limits
```

Remediation:

Every deployment may have unique requirements and settings. Recommended thresholds and settings are particularly important for MongoDB deployments:

- f (file size): unlimited
- t (cpu time): unlimited
- v (virtual memory): unlimited [1]
- n (open files): 64000
- m (memory size): unlimited [1] [2]
- u (processes/threads): 64000

Restart the mongod and mongos instances after changing the `ulimit` settings to ensure that the changes take effect.

Default Value:

Not configured

References:

1. https://docs.mongodb.com/manual/reference/ulimit/#recommended-ulimit-settings

6.4 Ensure that server-side scripting is disabled if not needed (Not Scored)

Profile Applicability:

- Level 2 - MongoDB on Linux

Description:

MongoDB supports the execution of JavaScript code for certain server-side operations: `mapReduce`, `group`, and `$where`. If you do not use these operations, server-side scripting should be disabled.

Rationale:

If server-side scripting is not needed and is not disabled, this introduces unnecessary risk that an attacker may take advantage of insecure coding.

Audit:

If server-side scripting is not required, verify that it is disabled (`javascriptEnabled` value of `false`) using the following command:

```
cat /etc/mongod.conf |grep –A10 "security" | grep "javascriptEnabled"
```

Remediation:

If server-side scripting is not required, disable it by using the `--noscripting` option on the command line.

Default Value:

Enabled

CIS Controls:

18.9 – Sanitize Deployed Software of Development Artifacts

For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.

6.5 Ensure The 'test' database is not installed (Not Scored)

Profile Applicability:

- Level 2 - MongoDB on Linux

Description:

The default MongoDB installation comes with an unused database called 'test'. It is recommended that the test database be dropped.

Rationale:

The test database can be accessed by all users and can be used to consume system resources. Dropping the test database will reduce the attack surface of the MongoDB server.

Audit:

Execute the following query to determine if the test database is present:

```
show dbs
```

Remediation:

Execute the following command mongoshell to drop the test database:

```
use test
```

```
db.dropDatabase()
```

CIS Controls:

18.9 – Sanitize Deployed Software of Development Artifacts

For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.

7 File Permissions

This section provides recommendations for setting permissions for the key file and the database file.

7.1 Ensure that key file permissions are set correctly (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

The key file is used for authentication in the sharded cluster. Implementing proper file permissions on the key file will prevent unauthorized access to it.

Rationale:

Protecting the key file strengthens authentication in the sharded cluster and prevents unauthorized access to the MongoDB database.

Audit:

To verify the permissions for the MongoDB key file, run the following command:

```
cat /etc/mongod.conf | grep "keyFile="
```

Remediation:

Set the `keyFile` ownership to mongodb user and remove other permissions by executing these commands:

```
chmod 600 /keyfile
sudo chown mongodb:mongodb /keyfile
```

Default Value:

Not configured

References:

1. https://docs.mongodb.com/v3.0/tutorial/enable-internal-authentication/

CIS Controls:

16.14 – Encrypt/Hash All Authentication Files and Monitor Their Access

Verify that all authentication files are encrypted or hashed and that these files cannot be accessed without root or administrator privileges. Audit all access to password files in the system.

7.2 Ensure that database file permissions are set correctly (Scored)

Profile Applicability:

- Level 1 - MongoDB on Linux

Description:

MongoDB database files need to be protected using file permissions.

Rationale:

This will restrict unauthorized users from accessing the database.

Audit:

To verify that the permissions for the MongoDB database file are configured securely, run the following commands.

Find out the database location using the following command:

```
cat /etc/mongod.conf |grep "dbpath"
```

Use the database location as part of the following command to view and verify the permissions set for the database file:

```
ls –l /var/lib/mongodb
```

Remediation:

Set ownership of the database file to mongodb user and remove other permissions using the following commands:

```
chmod 600 /var/lib/mongodb
sudo chown mongodb:mongodb /var/lib/mongodb
```

Default Value:

Not configured

CIS Controls:

14.4 – Protect Information with Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Appendix: Summary Table

Control		Set Correctly	
		Yes	No
1	**Installation and Patching**		
1.1	Ensure the appropriate MongoDB software version/patches are installed (Scored)	☐	☐
2	**Authentication**		
2.1	Ensure that authentication is enabled for MongoDB databases (Scored)	☐	☐
2.2	Ensure that MongoDB does not bypass authentication via the localhost exception (Scored)	☐	☐
2.3	Ensure authentication is enabled in the sharded cluster (Scored)	☐	☐
2.4	Ensure an industry standard authentication mechanism is used (Scored)	☐	☐
3	**Access Control**		
3.1	Ensure that role-based access control is enabled and configured appropriately (Scored)	☐	☐
3.2	Ensure that MongoDB only listens for network connections on authorized interfaces (Scored)	☐	☐
3.3	Ensure that MongoDB is run using a non-privileged, dedicated service account (Scored)	☐	☐
3.4	Ensure that each role for each MongoDB database is needed and grants only the necessary privileges (Scored)	☐	☐
3.5	Review User-Defined Roles (Scored)	☐	☐
3.6	Review Superuser/Admin Roles (Scored)	☐	☐
4	**Data Encryption**		
4.1	Ensure TLS or SSL protects all network communications (Scored)	☐	☐
4.2	Ensure Federal Information Processing Standard (FIPS) is enabled (Scored)	☐	☐
5	**Auditing**		
5.1	Ensure that system activity is audited (Scored)	☐	☐
5.2	Ensure that audit filters are configured properly (Scored)	☐	☐
5.3	Ensure that logging captures as much information as possible (Not Scored)	☐	☐
5.4	Ensure that new entries are appended to the end of the log file (Not Scored)	☐	☐
6	**Operating System Hardening**		
6.1	Mongodb Database Running with Least Privileges (Scored)	☐	☐
6.2	Ensure that MongoDB uses a non-default port (Scored)	☐	☐

6.3	Ensure that operating system resource limits are set for MongoDB (Not Scored)	☐	☐
6.4	Ensure that server-side scripting is disabled if not needed (Not Scored)	☐	☐
6.5	Ensure The 'test' database is not installed (Not Scored)	☐	☐
7	**File Permissions**		
7.1	Ensure that key file permissions are set correctly (Scored)	☐	☐
7.2	Ensure that database file permissions are set correctly (Scored)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
06-06-2017	1.0.0	Initial Release