UNIVERSITÄT
DES
SAARLANDES

CISPA
Center for IT-Security, Privacy
and Accountability

MongoDB databases at risk
–
Several thousand MongoDBs without access control on the Internet

Jens Heyens, Kai Greshake, Eric Petryka

January 2015

1 Introduction

MongoDB is an open source NoSQL database. It is currently the most commonly used NoSQL database and several major Web sites and services integrate such a database for their backend.

We discovered that MongoDB databases running as a service or Web site backend on several thousand commercial servers are openly available on the Internet. Without any special tools and without circumventing any security measures, we would have been able to get read and write access to thousands of databases, including, e.g., sensitive customer data or live backends of Web shops. The reason for this problem is twofold:

- The defaults of MongoDB are tailored for running it on the same physical machine or virtual machine instances.
- The documentations and guidelines for setting up MongoDB servers with Internet access may not be sufficiently explicit when it comes to the necessity to activate access control, authentication, and transfer encryption mechanisms.

If a less experienced administrator sets up a MongoDB Web server following those guidelines, it can easily happen that the administrator oversees the importance of activating crucially required security mechanisms. This will lead to a completely open and vulnerable database that each and everyone can access and, even worse, manipulate. Apparently, this is what happened to the thousands of databases we found. In the following we will first document our findings and, consequently, provide guidelines on how to secure MongoDB servers.

2 Finding MongoDB services

MongoDB runs by default on TCP port 27017, so an attacker would simply need to run a port scan on the Internet to find openly accessible databases. This is today incredibly easy and can be achieved within hours [4]. Even simpler for not so tech-savvy attackers is to identify accessible MongoDBs at the IoT search engine Shodan [1]. This Web service has a database containing IP addresses with a list of services running and an easy to use filter mask. Using a free standard account we identified a first set of vulnerable MongoDB addresses by pasting the following HTML code.

```
curl $SHODANURL |grep -i class=\"ip\" |cut -d '/' -f 3 \
|cut -d '"' -f 1|uniq >db.ip
```

3 Accessing MongoDB services

Essentially, we are already finished, Since we now are able to connect to the MongoDBs found by calling the mongo shell with the IP address found.

```
mongo $IP
```

4 Findings and responsible disclosure

In order to verify the impact and risk related to the found MongoDB instances, we exemplarily double-checked that these databases are not intentionally configured without access control and further security mechanisms. Briefly looking at a large database[1], we found a customer database of a French telecommunications provider with about 8 million customer entries.

Due to this finding, we decided to inform the competent data protection agency in France, CERTs and MongoDB Inc., such that the large amount of affected database owners can be both identified and notified. Our initial port scan revealed 39,890 instances. However, this number might be inaccurate, since on the one hand many larger providers blocked the scan such that there might be more publicly accessable MongoDBs online, and on the other hand some of these databases might be intentionally configured without security measures, e.g. as honeypots.

5 Proposed solution

The MongoDB service default configuration enables local access only. Its main configuration file is usually found at:

- Linux: /etc/mongodb.conf
- BSD: /usr/local/etc/mongodb.conf
- Windows: no default path; usually assigned on setup

Many precompiled packages of MongoDB already ship with a default configuration that binds the service after its installation only to localhost:

[1] The initial handshake required to find publicly accessible MongoDBs includes meta information such as the size and various other server settings

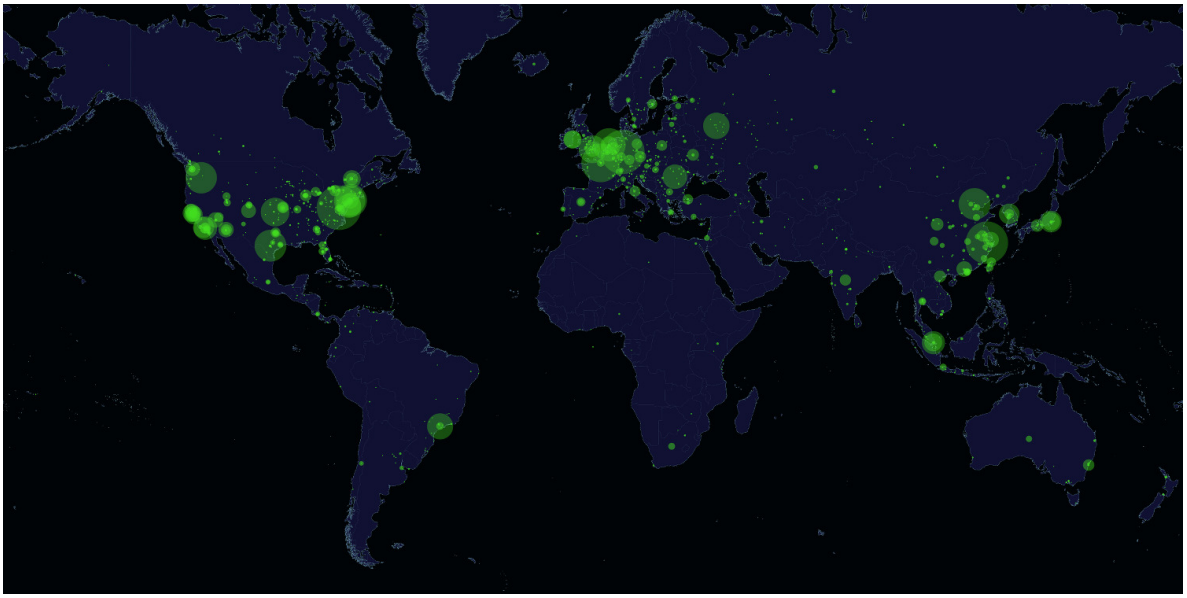

Figure 1: World-wide distribution of openly accessible MongoDBs

```
bindIp: 127.0.0.1
port: 27017
```

This allows access from services running on the same physical or virtual host and denies everything else. No other security feature like traffic encryption or access control is enabled by default. This configuration is acceptable, if the use-case scenario includes only services that need to access MongoDB from the same host as the database service.

However, a common setup and scalable solution for most Internet services is to have a database server running on one physical machine, while the services using this database service are (often virtualized) running on another machine. In this case, the easiest solution is to comment out the flag «bind_ip = 127.0.0.1» or to remove it completely[2], which defaults to accepting all network connections to the database. If access is possible from untrusted machines (e.g., from the Internet) outside the trusted network, it is crucial to also set up transfer encryption and proper access control.

5.1 Securing MongoDB from unauthorized access

To secure a running MongoDB instance from unauthorized access you basically need to enable

1. Access Control
2. Traffic/Transport Encryption

In this document, we will not explain in detail *how* you set either one up, but briefly explain the possible options, while giving the apropriate links to the MongoDB developer's documentation.

[2]Ideally, bindIp is at least limited to a set of internal network addresses

5.1.1 Access control

MongoDB supports four different types of user authentication, enabling it to be properly integrated into existing authentication mechanisms:

1. Challenge and Response (MONGODB-CR)
2. X.509 Certificate Authentication
3. Kerberos Authentication
4. LDAP Proxy Authentication

In all cases user credentials need to be added to MongoDB. Please refer to [5] for a how-to.

MONGODB-CR This authentication mechanism is MongoDB's default one. ≪Challenge and Response≫ is essentially the method most operating systems use to handle their user authentication: A user needs to know his username and a matching password. See [9] for more information

X.509 MongoDB supports an X.509 certificate-based client authentication[3]. X.509 is a widely used standard for a public-key-infrastructure [2, 3] (e.g., in SSL/TLS). In case your institution or company has already set up such a PKI for authentication of other services, MongoDB can be configured to use the existing infrastructure (cf. [11]).

Kerberos Kerberos is an authentication protocol for large network infrastructures. To use MongoDB with Kerberos you need a MongoDB Enterprise license (cf. [7]).

LDAP LDAP is a directory access protocol often used in UNIX/Linux/BSD environments (partly) to authenticate users. It is compatible with Microsoft's Active Directories (AD). To use MongoDB with LDAP, you need a MongoDB Enterprise for Linux license which supports LDAP authentication with an Active Directory. Currently, MongoDB Enterprise for Windows does not support LDAP (cf. [8]).

5.1.2 Traffic/transport encryption

A properly configured traffic encryption is needed to ensure that requested documents out of MongoDB and the previously configured access control are not visible to devices between the MongoDB host and the service hosts connecting to it. MongoDB supports SSL/TLS for this purpose which is the standard for encrypting sensitive traffic. It is recommended to only use strong SSL ciphers. MongoDB v2.6 upwards only allows strong SSL ciphers. See [10] and [6] for more information.

[3] Details: `http://docs.mongodb.org/manual/tutorial/configure-x509-client-authentication/`

References

[1] Shodanhq. Available at https://www.shodan.io/. Accessed: 2015-01-20.

[2] M. Cooper, Y. Dzambasow, P. Hesse, S. Joseph, and R. Nicholas. RFC 4158 - Internet X.509 Public Key Infrastructure: Certification Path Building. Online at ftp://www.ietf.org/rfc/rfc4158.txt, 2005.

[3] M. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Online at ftp://www.ietf.org/rfc/rfc5280.txt, 2008.

[4] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Zmap: Fast internet-wide scanning and its security applications. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 605–620, Berkeley, CA, USA, 2013. USENIX Association.

[5] MongoDB-Documentation. Add a User to a Database. Available at http://docs.mongodb.org/manual/tutorial/add-user-to-database/. Accessed: 2015-01-30.

[6] MongoDB-Documentation. Configure mongod and mongos for SSL. Available at http://docs.mongodb.org/manual/tutorial/configure-ssl/. Accessed: 2015-01-30.

[7] MongoDB-Documentation. Kerberos Authentication. Available at http://docs.mongodb.org/manual/core/authentication/#security-auth-kerberos. Accessed: 2015-01-30.

[8] MongoDB-Documentation. LDAP Proxy Authority Authentication. Available at http://docs.mongodb.org/manual/core/authentication/#security-auth-ldap. Accessed: 2015-01-30.

[9] MongoDB-Documentation. MONGODB-CR Authentication. Available at http://docs.mongodb.org/manual/core/authentication/#authentication-mongodb-cr. Accessed: 2015-01-30.

[10] MongoDB-Documentation. Upgrade a Cluster to Use SSL. Available at http://docs.mongodb.org/manual/tutorial/upgrade-cluster-to-ssl/. Accessed: 2015-01-30.

[11] MongoDB-Documentation. X.509 Certificate Authentication. Available at http://docs.mongodb.org/manual/core/authentication/#security-auth-x509. Accessed: 2015-01-30.