In this thread I would like to discuss MongoDB Database Security Best Practices. I will talk about the best practices for securing MongoDB database users as well as queries. I will discuss securing data in a MongoDB database. I will define encryption at rest and during transit. As well, MongoDB’s recommended password policy will be explained.

To secure database users in MongoDB it is important to “create a [user administrator](https://www.mongodb.com/docs/manual/tutorial/configure-scram-client-authentication/#std-label-create-user-admin) first, then create additional users. The we should create a unique MongoDB user for each person/application that accesses the system.” (*Security Checklist — MongoDB Manual*, n.d.) As well, we should “follow the principle of least privilege. Create roles that define the exact access rights required by a set of users. Then create users and assign them only the roles they need to perform their operations. A user can be a person or a client application.” (*Security Checklist — MongoDB Manual*, n.d.).

If you have the done the step above, you are on the right path to secure database queries. Using role-based access control and only allowing a system of privileges will govern what users can and can not do. “These privileges can restrict users to only reading or updating specific collections for example. The RBAC model provides a system of fine-grain control that, when done well, can go a long way towards protecting the contents of the database in the event of a user compromise.” (Dunham, 2019). This means that potential database abusers will not have access to be able to perform certain queries within the database.

The best way to secure your data in a MongoDB database is to encrypt your data. “In most data leaks, the data is readable by unauthorized users. Encrypting data makes it unreadable by those who do not have the keys to decrypt it.” (*Security Checklist — MongoDB Manual*, n.d.). “Data at rest is data that is stored in one location where it can be protected and monitored. Encryption at rest is when data at rest is given layers of encryption for security. For instance, let’s consider bank account information in its original form a potential security risk. However, this sensitive data can be transformed into another unrecognizable form of data that must be encrypted to be accessed. The algorithm for the encryption is handled by an encryption key which only you, or authorized persons, obtain, meaning that other hackers will not have the means to decipher the encrypted data. As a result, should someone gain access to your data, they will be unable to get past the layers of encryption without an encryption key, and your data remains unreachable.” (*Encryption at Rest: Its Purpose and Benefits/Setbacks*, 2021). For data in transit, “MongoDB Atlas uses encryption in transit from application client to server and within intra-cluster communications by setting a set of [certificates](https://docs.mongodb.com/manual/tutorial/configure-ssl/#certificate-authorities) for the servers. MongoDB Atlas uses “[Let’s Encrypt”](https://letsencrypt.org/)known certificates to authenticate TLS (Transport Layer Security) enabled clients once they pass access and authentication controls.” (MongoDB, n.d.).

The “password policy” for MongoDB utilizes SCRAM. Salted Challenge Response Authentication Mechanism (SCRAM) is the default authentication mechanism for MongoDB. When a user [authenticates](https://www.mongodb.com/docs/manual/tutorial/authenticate-a-user/#std-label-authentication-auth-as-user) themselves, MongoDB uses SCRAM to verify the supplied user credentials against the user's [name](https://www.mongodb.com/docs/manual/reference/system-users-collection/#mongodb-data-admin.system.users.user), [password](https://www.mongodb.com/docs/manual/reference/system-users-collection/#mongodb-data-admin.system.users.credentials) and [authentication database](https://www.mongodb.com/docs/manual/reference/system-users-collection/#mongodb-data-admin.system.users.db) SCRAM is based on the IETF [RFC 5802](https://tools.ietf.org/html/rfc5802) standard that defines best practices for the implementation of challenge-response mechanisms for authenticating users with passwords. (*Security Checklist — MongoDB Manual*, n.d.).

References:

Dunham, C. (2019, August 23). *Top 10 security best practices for MongoDB*. Sqreen Blog. Retrieved November 28, 2022, from <https://blog.sqreen.com/top-10-security-best-practices-for-mongodb/>

*Security Checklist — MongoDB Manual*. (n.d.). Retrieved November 28, 2022, from <https://www.mongodb.com/docs/manual/administration/security-checklist/>

*Encryption at rest: Its purpose and benefits/setbacks*. (2021, June 28). Catalyst2. Retrieved November 28, 2022, from <https://www.catalyst2.com/blog/encryption-at-rest-its-purpose-and-benefits-setbacks/>

MongoDB. (n.d.). *MongoDB Data Encryption*. Retrieved November 28, 2022, from <https://www.mongodb.com/basics/mongodb-encryption>