Secure Use of Mongoose

In this document, we discuss how the Mongoose Object Data Model can be used securely with MongoDB.

Types of Attacks on NoSQL Databases:

- **NoSQL Injection:** A type of injection attack that exploits vulnerabilities in NoSQL Databases. It allows unauthorized actors to bypass authentication and authorization and access and modify database contents. Eg - accessing password hashes of all users. [1]
- **JSON Injection**: JSON Injection attacks in web applications are cyber attacks that seek to inject malicious code into an application to alter its normal execution. This can allow an attacker to insert extra fields in the documents, as Mongoose doesn't run validation by default on Update(). [3]

These attacks can be solved by the following measures

- **Input Validation** specifically means format validation ,i.e., checking that data is in the right format (Eg - email has only email, name has only letters, phone number has only numbers). [1][3]
- **Input Sanitization** means removing unwanted characters and leading/trailing white spaces, converting data to the right type. Eg - sanitizeFilter() in mongoose, toInt(), toFloat(), normalizeEmail(), whitelist(), etc. in **express-validator**. [1][2][3][5]

Mongoose conducts validation only on the following 2 occasions:

- When using the **save()** method to create new documents or to update existing documents. [4]
- When using the **create()** method to create new documents. [4]

Prevention:

- Mongoose doesn't perform validation by default when running the **updateOne(), updateMany()** or **findOneAndUpdate()** methods. To enable validation while running these methods, we must set the **runValidators** property to true. [4]
- It is also possible to run validation manually using the **validate()** or **validateSync()** methods. These methods can be used to validate a newly-created document. [4]
- In terms of sanitization, it is possible to prevent NoSQL Injection attacks by using the **sanitizeFilter()** method. [2]

Why We Need To Use Express-Validation:

- The Mongoose built-in validator only has the **Type, Required, min, max, MinLength, MaxLength, Match, Enum** and **Unique** validators available. [4]
- It doesn't have format validators for emails, dates, phone-numbers, regex for passwords, etc. While custom validators can be defined, **express-validator's** validators have been defined by professionals and are rigorous. [4]
- Before running validation, mongoose tries to coerce the data in the request body given to it to the correct data types in the schema. This process is called casting and if it cannot do that, then it gives us a **Casting Error**. To prevent this, it is better to validate the data with the **express-validator** library. [2][4]

REFERENCES

[1] https://www.strongdm.com/what-is/nosql-injection
[2] https://thecodebarbarian.com/whats-new-in-mongoose-6-sanitizefilter.html
[3] https://www.comparitech.com/net-admin/json-injection-guide/
[4] https://mongoosejs.com/docs/validation.html