

Post API VS Get API thought process.

→ The thought process of Get API is different from Post API

Brief information.

Whenever you look API we look it at the level that the API is Post API, or Get API.

Post API

→ what happens in Post API (think like you are the guard of the database.)

What is the job of the guard

~~The Guard~~ Guard of the Security

The job of the guard is, not to let in any random people inside only authorized people can go inside. and if someone is going outside from inside it makes sure that no data is leaking.

Post API means user is trying to enter data to database.

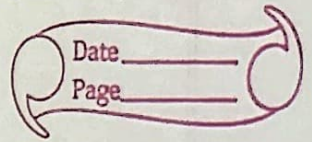
Get API means user is trying to fetch the data from the database.

How can attacker exploit a Post API

— They can attack your Post API by sending some random data into your API and we mistakenly put that data into database.

(If we put data that we don't want to, that's the opportunity of attacker that's something we don't want to do as a Guard of the database.)

Post threat \rightarrow attacker ^{can} send malicious data to our database
get threat \rightarrow attacker can ^{can} log in or get unauthorized data.



~~whenever~~

For POST API \rightarrow Always think that attacker can put anything to our database, and ~~have~~ we should have OAD that we will verify everything which is coming it to the request.

(Sanitize) \downarrow
(validate) \downarrow
(do all checks) \downarrow
(calculate) \downarrow
(save()) \downarrow
(send res)

connectionData.save(); \leftarrow it should be the second last line and we have to sanitize and ~~verify~~ validate everything before doing this

do all checks

Thought Process Change very different at get API

Saving

Here we are not ~~passing~~ saving any information to our database, here we are getting information from database or we are fetching data for the user

In get APIs we will 100% make sure that we are sending only allowed data. (Make sure user is verified user and whatever he is requesting is allowed for his scope)

Always make sure what you are sending back to the client is 100% authorized and is in the user scope.

const hideUserFromFeed = new Set();

Set() \Leftarrow is like an array but if we put duplicate values it will ignore the duplicate values and always have unique values

① find all the received or send request which user has done.

② we will find unique Id \Leftarrow create Set and Push to call to Set

const user = await user.find({

\$and: { \$id: { \$in: Array.from(hideUserFromFeed) },
~~\$or~~ \$id: { \$ne: { \$eq: loginUser._id } }

\$and \$or \$nin \Leftarrow not in \$ne \Leftarrow not equal to.

.select(); \Leftarrow only select the data only data we want.

Pagination.

Q we have created all the ops now suppose we have 1000 user then our find should not load 1000 user data. it should load in 10, 10, 10 ... style.

1

for this we will use pagination.

best way is using query params.

skip(0) & limit(10)

find ? page = 1 & limit = 10 \Rightarrow first 10 user 1-10

find ? page = 2 & limit = 10 \rightarrow 11-20 skip(10) & limit(10)

find ? page = 3 & limit = 10 \rightarrow 21-30

skip(20) & limit(10)

in mongoDB we have two function

start

①. skip() \leftarrow How many doc you want to skip from the 1st

②. limit() \leftarrow how many document do you want

and it skip(0) & limit(10) \rightarrow give first 10

$$\text{Skip} = (\text{page} - 1) * \text{limit}$$

• skip(skip)

let limit = parseInt(req.query.limit) || 5;

~~limit = Math.min~~

limit = Math.min(limit, 10)

\uparrow Serialized the data

max limit is 10

Completed All backend work
for Trade Circle



Now it's time to create frontend.



All frontend notes are on FrontEnd (2)
book, in first book we have different
Project in (2) book we will have
Trade Circle frontend Project.