# Restrict api access in Node JS express

Asked 4 years, 10 months ago  Modified 4 months ago  Viewed 9k times

7

I have an express server with a few API routes like this:

```
server.post("/api/send-email", (req, res) => {

   });
});
```

You don't need an auth token to access the API, but I only want my website `mydomain.com` to be able to use it. I have tried enabling restricting access like this:

```
function restrictAccess(req, res, next) {
      if (req.headers['origin'] !== 'http://localhost:3000') {
        res.sendStatus(403);
      } else {
        next();
      }
    }
```

And I then passed restrict access into my route as a middleware. When I make a POST request with postman I can't the API anymore, but I can just change the origin header and am able to access it again.

How can I allow only requests from `mydomain.com`? I have searched the internet for a long time now, but couldn't find anything. Is it even possible?

node.js  express  cors

Share  Improve this question  Follow

asked Feb 22, 2020 at 17:38

Jona

315 ● 1 ● 3 ● 10

1 In a nutshell, you can't. Any tool like postman or any script can't send whatever headers it wants so headers by themselves are useless for restricting access to only a web page in your domain. That's just how the web works. Restricting access would more commonly require a user login or something like that (probably setting a cookie that you can check) and then if you see that your APIs are being abused, you can ban/remove a specific user account that is doing it. – jfriend00 Feb 22, 2020 at 18:14

2 Answers

Sorted by: Highest score (default)

12

> How can I allow only requests from my own webpages at mydomain.com?

**In a nutshell, you can't.** Any tool like postman or any script (such as node.js, PHP, Perl, etc...) can send whatever headers or other request parameters it wants so headers by themselves are useless for restricting access to only a web page in your domain.

That's just how the web works.

Restricting access would more commonly require a user login or some credential like that (probably setting a cookie that you can check) and then if you see that your APIs are being abused, you can ban/remove a specific user account that is doing it.

There are other techniques that may make it more work for scripts or tools to use your API, but even they are not immune to a hacker that wants to put in the work. For example, your server can generate a token, embed it in your web page and then whenever you make an API request from your web page, you include the token from the web page. Your server, then checks for the presence of a valid token. You make sure that tokens expire in some reasonable amount of time so a hacker can't just get one and use it for a long time.

A determined hacker can still scrape a token out of your web page whenever they want to use it so this is only an obstacle, not something that stops a determined hacker.

The only real solution here and what someone like Google uses for their APIs is to require some sort of credential with each API call and then instrument your server for abuse of the APIs (rate limiting, unintended use, etc...) and then revoke that credential if it is being misused. The credential can be a developer token (as with some Google APIs) or it can be some sort of authentication credential from a user login (like perhaps a cookie).

There are other tricks I've seen used before where an API only works properly if a sequence of requests came before it that would normally be coming from your web page. This is a lot more work to implement and maintain, but if your web page would normally issue a request for the web page, then make two ajax calls, then request five images and then call the API, you can somehow have your server track this sequence of events from a specific browser and only if you see the expected sequence of events that looks like it's coming from a real browser web page, so you allow the API call to work. Again, this is a lot of work and still not infallible because determined hacker can just use something like puppeteer to automate an actual browser.

Share

Improve this answer

Follow

edited Apr 9, 2022 at 8:05

answered Feb 22, 2020 at 18:23

jfriend00

**706k** ● 103 ● 1k ● 1k

---

1

Major browsers send along the origin header without permitting any browser Javascript to modify it.

Non-browser API clients, like Postman and anything else, can set the origin header, and other headers, to whatever they choose. Non-browser API clients can easily spoof your API pretending to be browsers.

Therefore, **security tip,** using the origin header's value to decide whether to grant access to your API offers you no security whatsoever.

You really do need some kind of token access mechanism. Especially for an API that sends email. If a cybercreep finds your API, your hosting service will accuse you of sending spam.

Sorry about that. :-( Security is a pain in the neck.

Share Improve this answer Follow

answered Feb 22, 2020 at 17:57



O. Jones

**108k** ● 17 ● 128 ● 180