EcomLabs

STEP-BY-STEP GUIDE

Review your NEXT_PUBLIC_ variables

A five-step review to find secrets in public Next.js variables and confirm in the build that they do not reach the browser, plus the steps to rotate a key that already leaked.

FOR	Next.js projects on the App Router
TESTED WITH	Next.js 15.5.23 · React 19.0.0 · Node 22 · Windows 11
UPDATED	October 30, 2026
DOCS	nextjs.org/docs/app/guides/environment-variables
ARTICLE	ecomlabs.dev/g/secretos-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you will review
Step 01	List your public variables
Step 02	Classify each variable
Step 03	Search the build for your secrets
Step 04	Move the secret to the server
Step 05	Review your client components
If it leaked	Rotate the key
Final	Checklist

BEFORE YOU START

What you will review

In Next.js, variables with the `NEXT_PUBLIC_` prefix are copied as plain text into the JavaScript the browser downloads when you run `next build`. The rest only exist on the server. This review looks for secrets with the wrong prefix, checks the build for what reached the browser, and moves whatever needs it to the server.

The commands are for Git Bash, macOS or Linux. The results shown come from a demo app on Next.js 15.5.23 with fake values such as `sk_test_FALSO_PUBLICO_123` (`FALSO` is Spanish for fake).

> **CAUTION**
> Do not paste full keys into a terminal or a chat: they stay in the history. To search for them, use only their first few characters, and never print your `.env` files whole.

STEP 01

List your public variables

First, the names your code uses. Adjust the folders to your project:

```
grep -rhoE "NEXT_PUBLIC_[A-Z0-9_]+" app components lib | sort -u
```

Then, the names defined in your `.env` files. This command prints only the name, never the value:

```
grep -ho "^NEXT_PUBLIC_[A-Z0-9_]*" .env*
```

In the demo app, `NEXT_PUBLIC_PAYMENTS_SECRET_KEY` showed up. A name with `SECRET`, `TOKEN`, `PASSWORD` or `PRIVATE` next to the public prefix is already a red flag. Check your host's environment variables too: there may be one that is not in your local files.

STEP 02

Classify each variable

The question for each one: if it were published tomorrow, would you have to rotate it? If yes, it gets no prefix.

KIND OF VALUE	CAN IT TAKE NEXT_PUBLIC_?
Store name, public URL of the site or the API	Yes
GA4 or Google Tag Manager ID	Yes
Publishable or client key the provider designates for the browser	Yes, with whatever restrictions the provider allows
Payments secret key	Never
Database URL with a username and password	Never
Admin API tokens (Shopify Admin, CMS, ERP)	Never
Secrets that sign webhooks or sessions	Never
Keys for pay-per-use services (AI, email)	Never

STEP 03

Search the build for your secrets

Build the project and search the folder served to browsers for the first characters of each secret key. In the demo app:

```
grep -rlF "sk_test_FALSO_PUB" .next/static
```

The result was `.next/static/chunks/app/bad/page-78203093a6f45326.js`: the key sat in a file anyone can download. Search the prerendered HTML too, where an unprefixed secret can end up if you pass it as a prop or render it in a client component:

```
grep -rlF "sk_test_FALSO_SERVIDOR" .next/server/app
```

In the demo app, `leak.html`, `leak.rsc` and `probe.html` came up, the two pages we got wrong on purpose. In your project, both commands should come back empty.

STEP 04

Move the secret to the server

Drop the prefix, rename it in your code and keep all access to the key in one server module. This is the top of `lib/payments.js` in the demo app:

```
import "server-only";

export async function createSession() {
  const res = await fetch(`${process.env.PAYMENTS_API_URL}/sessions`, {
    method: "POST",
    headers: { Authorization: `Bearer ${process.env.PAYMENTS_SECRET_KEY}` },
  });
  if (!res.ok) throw new Error(`payments: ${res.status}`);
  return res.json();
}
```

`import "server-only"` makes the build fail if a client component imports this module. The browser calls your server instead, for example through this Route Handler, `app/api/checkout/route.js`, which returns only the session id:

```
import { createSession } from "@/lib/payments";

export async function POST() {
  const session = await createSession();
  return Response.json({ id: session.id });
}
```

From your own UI you can also use a Server Action with `"use server"` that calls `createSession()` and returns only the id.

STEP 05

Review your client components

List the `'use client'` files that read `process.env`:

```
grep -l "process\.env" $(grep -rlE "^[\"']use client[\"']" app components)
```

In each result, the only valid read is a `NEXT_PUBLIC_` variable with a public value. Also check which props your client components receive from the server: a secret passed as a prop travels in the HTML even if the component never shows it.

If you changed a `NEXT_PUBLIC_` value on your host, rebuild: it is inlined at build time, and the site keeps the old value until the next build.

IF IT LEAKED

Rotate the key

If a secret key reached production with the public prefix, treat it as compromised. Dropping the prefix does not invalidate it: the old JavaScript lives on in caches, CDNs and downloaded copies.

1. **Create a new key** in the provider's dashboard, with the minimum permissions your server needs.

2. **Store it without the prefix** in your host's environment variables and do steps 4 and 5.
3. **Build and deploy.** Repeat step 3 with the first characters of the old key: it must not show up.
4. **Revoke or delete the old key** at the provider. This is the step that closes the leak.
5. **Check the provider's logs** since the day the key reached production: charges, refunds or reads you do not recognize.
6. **Check the repository.** If the key ever made it into a commit, it stays public even after you delete the file. Make sure your `.env` files are in `.gitignore`.

> **CAUTION**
>
> If the key moves money or customer data, do not wait for the deploy: revoke it first and accept a few minutes of checkout errors.

FINAL

Checklist

- [ ] No `NEXT_PUBLIC_` variable holds a secret key, a token or a password.
- [ ] Your host's variables match what you reviewed locally.
- [ ] `grep` finds none of your secret keys in `.next/static` or `.next/server/app`.
- [ ] Modules that use secrets start with `import "server-only"`.
- [ ] No client component reads a secret or receives one as a prop.
- [ ] Route Handlers and Server Actions return only what the UI needs.
- [ ] If a key was exposed, you rotated it, revoked it and checked the logs.