EcomLabs

GUÍA PASO A PASO

Revisa tus variables NEXT_PUBLIC_

Una revisión en cinco pasos para encontrar secretos en variables públicas de Next.js y comprobar en el build que no llegan al navegador, más los pasos para rotar una clave que ya se filtró.

PARA	Proyectos Next.js con App Router
PROBADO CON	Next.js 15.5.23 · React 19.0.0 · Node 22 · Windows 11
ACTUALIZADA	30 de octubre de 2026
DOCUMENTACIÓN	nextjs.org/docs/app/guides/environment-variables
ARTÍCULO	ecomlabs.dev/g/secretos
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué vas a revisar
Paso 01	Lista tus variables públicas
Paso 02	Clasifica cada variable
Paso 03	Busca tus secretos en el build
Paso 04	Mueve el secreto al servidor
Paso 05	Revisa los componentes de cliente
Si ya se filtró	Rota la clave
Al terminar	Checklist

ANTES DE EMPEZAR

Qué vas a revisar

En Next.js, las variables con el prefijo `NEXT_PUBLIC_` se copian como texto en el JavaScript que descarga el navegador cuando haces `next build`. Las demás solo existen en el servidor. Esta revisión busca secretos con el prefijo equivocado, comprueba en el build qué llegó al navegador y mueve lo que haga falta al servidor.

Los comandos son de Git Bash, macOS o Linux. Los resultados que ves son de una app de prueba con Next.js 15.5.23 y valores falsos, como `sk_test_FALSO_PUBLICO_123`.

> **CUIDADO**
>
> No pegues claves completas en la terminal ni en un chat: quedan en el historial. Para buscarlas, usa solo sus primeros caracteres, y nunca imprimas tus archivos `.env` enteros.

PASO 01

Lista tus variables públicas

Primero, los nombres que usa tu código. Ajusta las carpetas a tu proyecto:

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

Después, los nombres definidos en tus archivos `.env`. Este comando imprime solo el nombre, nunca el valor:

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

En la app de prueba apareció `NEXT_PUBLIC_PAYMENTS_SECRET_KEY`. Un nombre con `SECRET`, `TOKEN`, `PASSWORD` o `PRIVATE` junto al prefijo público ya es una alarma. Revisa también las variables de entorno de tu hosting: ahí puede haber una que no esté en tus archivos locales.

PASO 02

Clasifica cada variable

La pregunta para cada una: si mañana aparece publicada, ¿tienes que rotarla? Si la respuesta es sí, no lleva el prefijo.

TIPO DE VALOR	¿PUEDE LLEVAR NEXT_PUBLIC_?
Nombre de la tienda, URL pública del sitio o de la API	Sí
ID de medición de GA4 o de Google Tag Manager	Sí
Clave publicable o de cliente que el proveedor indica para el navegador	Sí, con las restricciones que el proveedor permita
Clave secreta de pagos	Nunca
URL de la base de datos con usuario y contraseña	Nunca
Tokens de APIs de administración (Shopify Admin, CMS, ERP)	Nunca
Secretos para firmar webhooks o sesiones	Nunca
Claves de servicios que cobran por uso (IA, correo)	Nunca

PASO 03

Busca tus secretos en el build

Construye el proyecto y busca los primeros caracteres de cada clave secreta en la carpeta que se sirve al navegador. En la app de prueba:

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

El resultado fue `.next/static/chunks/app/bad/page-78203093a6f45326.js`: la clave estaba en un archivo que cualquiera puede descargar. Busca también en el HTML prerenderizado, donde un secreto sin prefijo puede acabar si lo pasas como prop o lo pintas en un componente de cliente:

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

En la app de prueba aparecieron `leak.html`, `leak.rsc` y `probe.html`, las dos páginas que hicimos mal a propósito. En tu proyecto, ambos comandos deben terminar sin resultados.

PASO 04

Mueve el secreto al servidor

Quita el prefijo, cambia el nombre en tu código y deja todo el acceso a la clave en un módulo de servidor. Este es el principio de `lib/payments.js` en la app de prueba:

```
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"` hace que el build falle si un componente de cliente importa este módulo. El navegador llama a tu servidor, por ejemplo con este route handler, `app/api/checkout/route.js`, que devuelve solo el id de la sesión:

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

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

Desde tu propia interfaz también puedes usar una server action con `"use server"` que llame a `createSession()` y devuelva solo el id.

PASO 05

Revisa los componentes de cliente

Lista los archivos con `'use client'` que leen `process.env`:

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

En cada resultado, la única lectura válida es una variable `NEXT_PUBLIC_` con un valor público. Revisa también qué props reciben tus componentes de cliente desde el servidor: un secreto pasado como prop viaja en el HTML aunque el componente no lo muestre.

Si cambiaste un valor `NEXT_PUBLIC_` en tu hosting, vuelve a construir: se incrusta en el build y el sitio sigue con el valor viejo hasta el siguiente.

SI YA SE FILTRÓ

Rota la clave

Si una clave secreta llegó a producción con el prefijo público, considérala comprometida. Quitar el prefijo no la invalida: el JavaScript viejo sigue en cachés, CDNs y copias descargadas.

1. **Crea una clave nueva** en el panel del proveedor, con los permisos mínimos que tu servidor necesita.

2. **Guárdala sin prefijo** en las variables de entorno de tu hosting y haz los pasos 4 y 5.
3. **Construye y despliega.** Repite el paso 3 con los primeros caracteres de la clave vieja: no debe aparecer.
4. **Revoca o elimina la clave vieja** en el proveedor. Este paso es el que cierra la fuga.
5. **Revisa los logs del proveedor** desde la fecha en que la clave llegó a producción: cargos, reembolsos o lecturas que no reconozcas.
6. **Revisa el repositorio.** Si la clave llegó a un commit, sigue siendo pública aunque borres el archivo. Confirma que tus archivos `.env` están en `.gitignore`.

> **CUIDADO**
>
> Si la clave mueve dinero o datos de clientes, no esperes al deploy: revócala primero y acepta unos minutos de error en el checkout.

AL TERMINAR

Checklist

- [ ] Ninguna variable `NEXT_PUBLIC_` tiene una clave secreta, un token o una contraseña.
- [ ] Las variables del hosting coinciden con lo que revisaste en local.
- [ ] `grep` no encuentra tus claves secretas en `.next/static` ni en `.next/server/app`.
- [ ] Los módulos que usan secretos empiezan con `import "server-only"`.
- [ ] Ningún componente de cliente lee un secreto ni lo recibe como prop.
- [ ] Route handlers y server actions devuelven solo lo que la interfaz necesita.
- [ ] Si una clave estuvo expuesta, la rotaste, la revocaste y revisaste los logs.