

SAMPLE

Sample report: fictional app, for illustration. "DemoTodo" and demo-todo-app.example don't exist, and no real site was scanned. All keys, IDs and paths below are made up.



Milo Kits · milokits.com

MILO'S LAUNCH CHECK · DEEP CHECK

Launch Check report for DemoTodo

URL checked	https://demo-todo-app.example
Also covered	api.demo-todo-app.example, /pricing, /login, /settings, /admin, /api/tasks, /api/checkout, /api/webhooks/stripe
Checked on	14 Oct 2026, 09:30 UTC (report delivered 15 Oct, 08:05 UTC)
Plan	Deep Check (\$79), includes one free re-check
Stack spotted	Vite + React, Supabase, Stripe, OpenAI (built with an AI coding tool)
Requests made	112, passive and read-only, rate-limited

58_{/100}

NOT LAUNCH-
READY YET

Fix the 2 "fix first" items and this goes to about **85**. Most of it is a one-afternoon job.

Hi Sam,

Here's what I found looking at the public side of DemoTodo. I sorted everything by what to fix first. Each fix comes as a prompt you can paste into Cursor, Claude Code, Lovable or Bolt. Review the diff before you deploy, and rotate any key that was public. When you're done, reply and I'll run your free re-check.

At a glance

2

Fix first (high)

3

Fix soon
(medium)

2

Nice to have
(low)

7

Looks good

Headline: Your OpenAI key is in the public JavaScript, and the tasks table can be read by anyone without logging in. Fix those two before you share the link anywhere. Your payments setup is in good shape.

Findings

1. OpenAI API key in your JavaScript bundle **FIX FIRST**
2. Supabase table tasks readable without logging in **FIX FIRST**
3. /admin loads for visitors who aren't signed in **FIX SOON**
4. API answers requests from any website (CORS) **FIX SOON**
5. Missing security headers (CSP, HSTS, frame protection) **FIX SOON**
6. Sign-in page says which emails have accounts **NICE TO HAVE**
7. Source maps are public **NICE TO HAVE**

HIGH Fix first

1. OpenAI API key in your JavaScript bundle

`https://demo-todo-app.example/assets/index-8c1e3b.js`

WHAT I SAW

A string starting with `sk-proj-` is baked into your frontend bundle (masked: `sk-proj-...x9Q2`). It's used by the "Suggest subtasks" button, which calls OpenAI directly from the browser.

WHY IT MATTERS

Anyone can open their browser's developer tools, copy the key and use it on your bill. Leaked keys are often found and abused within hours. OpenAI may also disable the key, which would break the feature for everyone.

PASTE THIS INTO YOUR AI CODER

```
My OpenAI API key is exposed in the frontend bundle (it's used by the "Suggest subtasks" feature). Please:
1. Create a server-side route (e.g. /api/suggest) that reads OPENAI_API_KEY from a server-only environment variable and calls OpenAI there.
2. Remove the key and any VITE_ / NEXT_PUBLIC_ variable that holds it from all client code and .env files that get bundled.
3. Make the frontend call /api/suggest instead, require a logged-in Supabase user, and add a simple per-user rate limit.
4. Show me the diff and tell me exactly where to set OPENAI_API_KEY on my host.
```

THEN CHECK

Rebuild and deploy, then search your live JS for sk-. It should be gone. **Rotate the key** in the OpenAI dashboard and set a monthly spend limit. The old key has already been public.

2. Supabase table `tasks` readable without logging in

```
https://<your-project>.supabase.co/rest/v1/tasks?select=*&limit=1
```

WHAT I SAW

Using the public "anon" key that's already in your site's code, an anonymous request for one row came back with data. Columns I saw: `id`, `user_id`, `title`, `notes`, `due_at`. I didn't keep any row contents. In the RLS policies you sent me (Deep Check), Row Level Security is switched off for `tasks` and `task_shares`.

WHY IT MATTERS

Right now anyone can read every user's to-dos and notes, and very likely edit or delete them too. People put personal things in to-do apps, so this is a privacy problem as well as a security one.

PASTE THIS INTO YOUR AI CODER

```
My Supabase tables `tasks` and `task_shares` are readable with the anon key (no login). Please:
1. Enable Row Level Security on both tables.
2. Add policies so SELECT / INSERT / UPDATE / DELETE on `tasks` only work when auth.uid() = user_id, and a shared task is readable only by users listed for it in `task_shares`.
3. Check every other table in the public schema and tell me which ones still have RLS off.
4. Give me the SQL to run in the Supabase SQL editor, plus a quick way to confirm an anonymous request now gets no rows.
```

THEN CHECK

Logged out, in a private window, the same REST URL should return an empty list `[]` or an error, not your data. Logged in as a test user, you should see only that user's tasks.

MEDIUM

Fix soon

3. `/admin` loads for visitors who aren't signed in **DEEP**

```
https://demo-todo-app.example/admin
```

WHAT I SAW

The admin dashboard renders its layout and calls `/api/admin/stats` for logged-out visitors. The "Admin" link is hidden in the menu, but the page itself has no server-side check. The stats API returned an error for me, so the data looks protected, but the page shouldn't load at all.

WHY IT MATTERS

If "admin only" is just a hidden button, one forgotten API route is all it takes for a stranger to act as an admin.

PASTE THIS INTO YOUR AI CODER

```
/admin and /api/admin/* are only protected by hiding the link in the UI. Please:  
1. Add a server-side check that the current Supabase user has an admin role (stored in a  
table or app_metadata, not in localStorage) for every /api/admin/* route.  
2. Redirect logged-out or non-admin visitors away from /admin before the page renders.  
3. Return 401/403 from admin APIs for anyone who isn't an admin.  
4. Show me the diff and how to test it with a normal user account.
```

THEN CHECK

Open /admin in a private window: you should be sent to /login. Logged in as a normal user, admin APIs should return 403.

4. API answers requests from any website (CORS) DEEP

```
https://api.demo-todo-app.example/api/tasks
```

WHAT I SAW

The API copies whatever Origin the browser sends back into Access-Control-Allow-Origin and also sends Access-Control-Allow-Credentials: true.

WHY IT MATTERS

Another website could make requests to your API from a signed-in user's browser and read the answers. It's a classic way for one bad link to leak someone's data.

PASTE THIS INTO YOUR AI CODER

```
My API at api.demo-todo-app.example reflects any Origin in Access-Control-Allow-Origin and  
sends Access-Control-Allow-Credentials: true. Please:  
1. Allow only my real front-end origins (https://demo-todo-app.example and my local dev URL)  
from an allowlist.  
2. Only send Allow-Credentials when the origin is on the allowlist.  
3. Show me the diff and a curl command to confirm a request with Origin:  
https://evil.example gets no CORS headers back.
```

5. Missing security headers

```
Response headers on https://demo-todo-app.example
```

WHAT I SAW

No Content-Security-Policy, no Strict-Transport-Security, and nothing stopping other sites from loading your pages in a frame.

WHY IT MATTERS

These headers are cheap seatbelts. A CSP limits the damage if a script injection bug ever slips in, HSTS stops browsers falling back to plain HTTP, and frame protection blocks "clickjacking" tricks.

PASTE THIS INTO YOUR AI CODER

```
My site is missing security headers. Please add these in my host config (netlify.toml /
_headers / vercel.json, whichever I use):
- Content-Security-Policy: a sensible starter for a Vite + React app that uses Supabase and
Stripe Checkout (include my Supabase project URL and js.stripe.com; no 'unsafe-eval')
- Strict-Transport-Security: max-age=31536000; includeSubDomains
- X-Content-Type-Options: nosniff
- Referrer-Policy: strict-origin-when-cross-origin
- Permissions-Policy: camera=(), microphone=(), geolocation=()
- frame-ancestors 'none' in the CSP
Show me the exact file changes, and tell me how to spot anything the CSP breaks in the
browser console.
```

LOW

Nice to have

6. Sign-in page says which emails have accounts DEEP

<https://demo-todo-app.example/login> → "Forgot password"

WHAT I SAW

Requesting a reset for an unknown address shows "No account with that email". (I only tried made-up addresses.)

WHY IT MATTERS

It lets anyone check whether a person uses your app. That's low risk for a to-do app, but it's an easy fix.

PASTE THIS INTO YOUR AI CODER

```
On my forgot-password form, show the same message whether or not the email exists: "If an
account exists for that email, we've sent a reset link." Do the same for any sign-up or
login errors that reveal whether an email is registered. Show me the diff.
```

7. Source maps are public

<https://demo-todo-app.example/assets/index-8c1e3b.js.map>

WHAT I SAW

The production build ships .map files, so your original source code (with comments) can be downloaded.

WHY IT MATTERS

It's not a hole by itself, but it makes everything above much easier to find for someone poking around.

PASTE THIS INTO YOUR AI CODER

```
Please turn off source maps in my production build (build.sourcemap: false in vite.config)
and make sure no .map files end up in the deployed output. Show me the change.
```

✓ Looks good

- ✓ No .env , .git or backup files exposed
- ✓ HTTP redirects to HTTPS, and the certificate is valid
- ✓ Stripe: only the publishable key is public (that's expected)
- ✓ Checkout prices are set on the server, not taken from the browser **DEEP**
- ✓ Stripe webhook rejects unsigned requests **DEEP**
- ✓ Session cookies use Secure, HttpOnly and SameSite
- ✓ No debug or test pages left on the live site

🔍 What I couldn't check from the outside

- ✓ Anything only visible after logging in, beyond the pages listed above
- ✓ Your server-only environment variables and logs
- ✓ Your own code or dependencies (I didn't have access, by design)
- ✓ Real attacks, password guessing or load testing (I don't do those)

Your free re-check: once you've pasted the fixes and deployed, reply to this email and I'll re-run every check and send you an updated score.



The honest bit. This is a best-effort, passive review of the publicly visible surface of the app, as it looked at the time of the check. It isn't a penetration test, a security audit or a certification, and it can't find every issue. A cleaner report doesn't mean an app is secure. Use the fix prompts at your own discretion and review every change before deploying. See the terms.

Milo

Milo Kits · milokits.com · hello@milokits.com

Quick Check or Deep Check?

This sample is a Deep Check. Findings marked **DEEP** are the extras a Quick Check doesn't cover.

What's included	Quick · \$29	Deep · \$79
Outside-in review of one live URL	✓	✓
Secrets in your public HTML and JavaScript	✓	✓
Supabase / Firebase exposure (read-only, using public keys already on your site)	✓	✓
Exposed files, admin and debug routes	✓	✓
Security headers, cookies, HTTPS	✓	✓
Summary score and copy-paste fix prompts for every finding	✓	✓
Report within 24 hours (web + PDF)	✓	✓
Up to 10 extra pages, subdomains or API routes	–	✓
Review of Supabase RLS policies or Firebase rules you send us	–	✓
Sign-in, password reset and payment flow review (from the outside)	–	✓
CORS and API route checks	–	✓
One free re-check after you fix things	–	✓