Eigenflow

GUIDE EIGENFLOW

La skill d'audit de sécurité de Cloudflare

Notre vrai test sur notre appli

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté AUDIT. Le dépôt est ici : https://github.com/cloudflare/security-audit-skill. On l'a lancée le 30 octobre 2026 sur le code de notre propre hub, et elle nous a mis sur une vraie faille. Voici la commande, la demande exacte, notre session minute par minute et ce qu'on en a fait.

À savoir avant de lire

- Test du 30 octobre 2026, de 13 h 36 à 13 h 59 : Claude Code 2.1.280 avec Claude Opus 5.5, sur l'abonnement Claude, sans clé d'API, skill au commit c1c8a8c (14 septembre 2026), encore le dernier le 4 novembre.

- Notre audit est en profil rapide et en une seule passe. Il ne remplace pas un audit complet, et encore moins un pentest.

- Le dépôt est sous licence MIT, avec 59 067 étoiles et 3 840 forks le 4 novembre. Eigenflow n'a aucun lien avec Cloudflare.

1. Ce que c'est

C'est un dossier de consignes que ton agent de code lit avant d'auditer ton projet. Cloudflare l'a publié le 18 juin 2026, avec un billet de blog qui raconte comment ils l'ont utilisé sur leur propre code : l'outil qui en est né a sorti 7 245 problèmes à traiter dans leurs dépôts.

Le déroulé tient en six phases, toujours dans le même ordre :

- **Reconnaissance** : des agents dressent la carte de ton appli (qui peut faire quoi, par où entrent les données, ce qui touche aux fichiers, à la base, aux autres services).

- **Chasse** : des agents chasseurs, chacun sur un domaine, cherchent à casser le code plutôt qu'à le relire.

- **Validation** : chaque piste passe à un agent neuf qui n'a pas chassé et qui essaie de prouver qu'elle n'existe pas.

- **Sortie structurée** : les pistes deviennent des fiches dans `findings.json`, vérifiées par deux petits programmes fournis avec la skill.

- **Revérification** : d'autres agents relisent les fiches finales.

- **Rapport** : `REPORT.md`, `FINDINGS-DETAIL.md` et `NEEDS-VALIDATION.md`, avec pour chaque problème la plus petite correction possible.

2. Ce qu'il te faut

- Claude Code avec un abonnement Claude. D'après son README, la skill marche aussi avec d'autres agents de code.

- Node.js 22.20 ou plus pour la commande d'installation : tape `node --version` pour vérifier.

- Le code de ton appli sur ton ordinateur.

La skill est gratuite, l'audit consomme ton quota. Le nôtre a lancé 18 agents en 23 minutes. Au prix de l'API, Claude Code l'a chiffré à 17,86 dollars. Avec l'abonnement, ça s'est imputé sur notre quota, sans facture en plus.

3. L'installer

Dans le dossier de ton projet :

> npx skills add https://github.com/cloudflare/security-audit-skill --skill security-audit

Chez nous, l'installation a pris 4 secondes. L'installeur affiche son bilan avant de copier quoi que ce soit : « Safe » pour Gen, « 0 alerts » pour Socket, « Med Risk » pour Snyk. Il a fini sur un conseil qu'on te répète : relis une skill avant de t'en servir, elle tourne avec tous les droits de ton agent. On l'a fait : 15 fichiers de consignes et deux petits programmes qui lisent des fichiers, sans accès réseau.

4. La demande exacte

Ouvre Claude Code dans le dossier qui **contient** ton projet, pas dans le projet lui-même, et colle :

> Fais un audit de sécurité complet du code dans mon-projet/ avec la skill security-audit, profil quick, et écris le rapport dans audit/run-1.

Trois détails comptent, on les a appris en lisant la skill :

- **« audit de sécurité complet »** : sans ces mots, la skill reste en mode conseil. Elle répond à ta question, sans lancer les six phases ni écrire de rapport.

- **Le dossier du rapport doit être hors de ton projet.** Sinon, la skill s'arrête et te demande un autre chemin.

- **Le profil.** `quick` fait une seule vague de chasse. `standard`, celui par défaut, suit tout le déroulé, avec plusieurs vagues et deux critiques de couverture. `deep`, pour les gros projets sensibles, relance des vagues jusqu'à ce qu'un critique ne trouve plus de trou.

5. Notre session, minute par minute

- 13:36 : Claude Code charge la skill et lit ses fichiers de méthode.

- 13:37 : quatre agents de reconnaissance partent en même temps.

- 13:44 : cinq chasseurs partent, un par domaine : fichiers, connexion et CSRF, données et RLS, chat et rendu, divers.

- 13:51 : les chasseurs ont rendu 8 pistes.

- 13:52 : un critique de couverture relit la carte et note ce qui n'a pas été couvert.

- 13:53 : huit vérificateurs, un par piste, essaient de démonter chacune.

- 13:59 : rapport écrit, les deux programmes de vérification passent.

6. Lire le rapport

Le rapport range chaque piste dans une de trois cases :

- **Confirmée** : la faille est prouvée, avec sa gravité et sa correction.

- **À valider** (needs validation) : le chemin dans le code est vérifié par un agent indépendant, mais un fait manque pour trancher. Pas de gravité.

- **Rejetée** : l'agent qui doutait a gagné.

Notre résultat : aucune faille confirmée, 8 pistes à valider, aucune rejetée. La skill refuse d'exécuter le code de ton appli sans un bac à sable qui coupe le réseau et limite la mémoire, le disque et le temps. Notre Mac ne fournissait pas tout ça : elle a donc vérifié en lisant le code, et a classé « à valider » ce qui demandait une exécution ou un fait absent du dépôt.

Ne t'arrête pas à « aucune faille confirmée ». Le rapport le dit lui-même : « Ce n'est pas une absence de risque. » Pour chaque piste, il indique ce que toi seul peux vérifier.

7. Ce qu'on en a fait

Notre piste n° 1 disait en substance : aucune route de l'API ne demande d'identification, et tout dépend de qui peut joindre le serveur. Le dépôt ne le montre pas, nous si. On a vérifié à la main, en lecture seule : notre hub était en ligne, et son API répondait à n'importe qui, sans mot de passe.

> avant : curl http://notre-serveur/api/leads → HTTP 200

> après : curl http://notre-serveur/api/leads → HTTP 401

On a corrigé le jour même : un mot de passe devant le hub, au niveau du serveur web (nginx), et la correction proposée par la skill pour la piste n° 2, qui contrôle un identifiant reçu avant de s'en servir comme nom de dossier. Les autres pistes sont dans notre liste, dans l'ordre de priorité du rapport.

8. Avant de la lancer chez toi

- **Une passe ne suffit pas.** Cloudflare l'écrit : un seul passage trouvait à peu près la moitié des failles trouvées en cumulant plusieurs passages. Relance la skill après tes corrections, elle relit les rapports précédents pour viser les trous. Nous, on n'en a fait qu'une.

- **Ne la lance jamais sur ce qui est en ligne.** Elle travaille sur le code. Ses consignes interdisent de sonder un site déployé, un service externe ou les données d'autres utilisateurs.

- **N'audite que ton code**, ou un code que tu as le droit d'auditer.

- **Ce n'est pas un pentest.** C'est une relecture outillée, qui te dit où regarder. Pour une appli qui manipule de l'argent ou des données sensibles, fais aussi appel à un professionnel.

Sources : dépôt cloudflare/security-audit-skill (README, SKILL.md, licence), billet « Build your own vulnerability harness » du blog Cloudflare (18 juin 2026), notre session du 30 octobre 2026.