

6 réflexes pour que tes clés *ne fuient jamais*

Clés OpenAI, Stripe, AWS, Supabase... Une clé d'API, c'est ta carte d'identité et ta carte bancaire en même temps. Coche chaque case une fois, ça prend moins d'une heure.

Vicentia
BOVO DIGITAL



57 M

données volées chez Uber via une clé AWS laissée dans un dépôt privé (2016)

2 min

avant qu'une clé publiée sur GitHub soit utilisée (Orca, 2023)

1 sur 5

sites « vibe codés » laisse lire un secret à n'importe quel visiteur (RedHunt Labs)

1 Sors tes clés du code

- ☐ Toutes les clés sont dans un fichier `.env`, aucune dans le code
- ☐ Le code les lit avec `process.env.OPENAI_API_KEY` (Node) ou `os.environ["OPENAI_API_KEY"]` (Python)
- ☐ Un `.env.example` sans valeurs, pour savoir quoi remplir

2 Ignore le `.env` avant le 1er commit

```
# .gitignore
.env
.env.*
!.env.example
```

- ☐ Vérifie : `git check-ignore -v .env` doit répondre
- ☐ Déjà commité ? `git rm --cached .env`, puis réflexe 6

3 Jamais dans le navigateur ni l'appli mobile

- ☐ L'appel à l'IA part de ton serveur (route API, fonction serverless), jamais de la page
- ☐ Aucune clé secrète dans une variable `NEXT_PUBLIC_`, `VITE_` ou `EXP0_PUBLIC_` : elles finissent dans le code envoyé au visiteur

OpenAI : ne déploie jamais ta clé côté client.

4 Active les protections de GitHub

- ☐ **Push protection** : Settings → Code security → bloque une clé reconnue avant l'envoi
- ☐ **Double authentication** sur ton compte : Settings → Password and authentication
- ☐ Pareil pour ton équipe et tes prestataires

Pense à Uber : le dépôt était privé, pas la double authentification.

5 Le moins de pouvoir possible

- ☐ Une clé par projet (et par environnement : dev, prod)
- ☐ Droits restreints : seulement ce dont l'appli a besoin
- ☐ Une date d'expiration quand le service le permet
- ☐ Une limite de dépenses mensuelle (OpenAI, Anthropic, Google Cloud...)

Pense à Microsoft : un lien en « contrôle total » jusqu'en 2051.

6 Prépare le plan « clé qui fuit »

- ☐ Tu sais où révoquer chaque clé (page « API keys » de chaque service)
- ☐ Tu sais qui prévenir si la clé est partagée
- ☐ Tu connais l'ordre ci-dessous, avant d'en avoir besoin

64 % des secrets fuités en 2022 marchaient encore 3 ans après (GitGuardian).

! Ta clé a fuité ? Ne commence pas par supprimer le fichier. Elle reste dans l'historique de Git.

1 Désactive-la

tout de suite, dans le tableau de bord du service

2 Crées-en une nouvelle

et range-la dans le `.env`

3 Regarde l'usage

journaux, facture, appels inconnus depuis la fuite

4 Nettoie l'historique

en dernier (git filter-repo ou BFG), puis préviens ton équipe

L'audit express de 5 minutes

à faire aujourd'hui

```
# cherche des clés dans tout l'historique du dépôt
git log -p --all | grep -nE "AKIA[0-9A-Z]{16}|sk_live_|sk-"
```

Puis ouvre ton site en ligne → clic droit → Inspecter → Sources → **Ctrl + Maj + F** et cherche `sk-`. Un résultat ? Réflexe 6.