sharp-edges

sharp-edges

> *Catalogue généré le 2026-05-11*

En une phrase

Repère les API ou les configurations qui ressemblent à des « pièges à doigts » : techniquement utilisables, mais conçues d'une manière où l'usage normal mène facilement à une faille de sécurité.

Quand l'utiliser

- Tu conçois une bibliothèque ou une API publique et tu veux qu'elle soit « sûre par défaut ».
- Tu audites une configuration où une option mal réglée peut tout faire sauter.
- Tu utilises ou tu évalues une bibliothèque crypto et tu te demandes si son usage « naturel » est sécurisé.
- Tu veux comprendre pourquoi des libraires connues (type JWT) ont mauvaise réputation en matière de pièges.

Comment l'invoquer

- **Slash command** : `/sharp-edges`.
- **Phrases déclencheurs (texte)** : "footgun", "misuse-resistant", "secure defaults", "API usability", "dangerous configuration".
- **Auto-invocation** : Sur demande explicite.

Description détaillée

Il existe deux philosophies pour concevoir une API. La première : « on offre toute la flexibilité, c'est au développeur de bien l'utiliser ». La deuxième : « on rend le chemin sécurisé tellement évident qu'on ne peut pas vraiment se tromper » — c'est le « pit of success ». Ce skill vérifie si une API tombe dans la première catégorie (auquel cas elle produira des bugs en masse) ou dans la deuxième.

Concrètement, il chasse plusieurs types de « pièges ». Les pièges de choix d'algorithme : laisser le développeur choisir entre RSA, HMAC, ou pire `none` (le célèbre piège JWT où un attaquant peut désactiver la vérification de signature). Les valeurs par défaut dangereuses : un timeout par défaut de 0 qui signifie « accepte tout », un mode debug activé par défaut. Les paramètres ambigus : `lifetime=0` veut-il dire « immédiat » ou « infini » ? Personne ne sait. Les API qui demandent au développeur de se rappeler de règles spéciales (« n'oublie pas d'appeler `verify()` après `decode()` ») au lieu de les imposer automatiquement.

Le skill rejette aussi les excuses classiques : « c'est documenté » (les devs ne lisent pas la doc sous pression), « les utilisateurs avancés ont besoin de flexibilité » (90 % des usages « avancés » sont du copier-coller), « c'est la responsabilité du dev » (non, c'est le designer qui a mis le piège). Il propose à la place des principes : choix sécurisé par défaut, primitives dangereuses cachées derrière une couche haut niveau, configurations dangereuses rejetées à la validation.

Pour aller plus loin

Pour les détails techniques, exemples et patterns spécifiques, voir le SKILL.md original.

Source

- **Plugin** : `trailofbits/sharp-edges`
- **Nom interne** : `sharp-edges`
- **Fichier** : `/home/thymon/.claude/plugins/cache/trailofbits/sharp-edges/1.0.0/skills/sharp-edges/SKILL.md`

Revision #2
Created 2026-05-11 21:20:03 UTC by thymon
Updated 2026-05-11 21:37:34 UTC by thymon