AI TOOLS WIKI · MÉMO

Security Guidance *mémo*

L'essentiel pour utiliser **Security Guidance**, le plugin officiel d'Anthropic qui vérifie la sécurité du code écrit par Claude Code. Il n'a pas de commandes à taper : tout passe par des hooks automatiques et des variables d'environnement.

Famille Revue, sécurité et création de skills · **Éditeur** Anthropic · **Licence** Anthropic Commercial Terms
Note du wiki 11/15 · **Commandes vérifiées le** 25 septembre 2026
Fiche, notes et avis aitoolswiki.viellard.fr/outils/revue-et-securite/security-guidance

Installation

```
/plugin install security-guidance@claude-plugins-official
```

Prérequis : Claude Code ≥ 2.1.144, Python 3.8+ dans le `PATH`, un accès API fonctionnel (abonnement, clé API ou fournisseur tiers).

Les trois couches (automatiques)

Couche	Quand	Ce qu'elle fait
1. Alertes par motif	À chaque `Edit` / `Write`	Rappel instantané sur ~25 motifs dangereux : `yaml.load`, `pickle.load`, `innerHTML` brut, secrets en dur…
2. Revue LLM du diff	À la fin de chaque tour de Claude	Envoie le diff à un modèle (Opus 4.7 par défaut) ; les problèmes graves sont renvoyés à Claude pour qu'il les corrige avant de te répondre.
3. Revue agentique du commit	À chaque `git commit` / `git push`	Un agent lit les fichiers liés pour suivre le flux de données : IDOR, contournement d'auth, SSRF sur plusieurs fichiers.

Réglages (variables d'environnement)

Variable	Effet
`SECURITY_GUIDANCE_DISABLE=1`	Désactive tout le plugin.
`ENABLE_PATTERN_RULES=0`	Désactive la couche 1.
`ENABLE_STOP_REVIEW=0`	Désactive la couche 2 seulement. Utile en multi-agents quand un autre agent change la branche entre deux tours.
`ENABLE_COMMIT_REVIEW=0`	Désactive la couche 3.
`ENABLE_CODE_SECURITY_REVIEW=0`	Désactive toutes les revues LLM (couches 2 et 3).
`SECURITY_REVIEW_MODEL=…`	Modèle de la couche 2 (ex. `claude-sonnet-4-6` pour réduire le coût).
`SG_AGENTIC_MODEL=…`	Modèle de la couche 3.
`SG_DUAL_OR=on`	Deux revues en parallèle et union des résultats : un peu plus de failles trouvées, coût ×2.

Sur Bedrock ou Vertex, utiliser l'identifiant propre au fournisseur (ex. `us.anthropic.claude-opus-4-7`), sinon la revue ne trouve jamais rien.

Règles propres à ton organisation

Créer un fichier `claude-security-guidance.md`, chargé dans cet ordre :

Emplacement	Portée
`~/.claude/claude-security-guidance.md`	Toutes tes sessions.
`<projet>/.claude/claude-security-guidance.md`	Le projet (à commiter).
`<projet>/.claude/claude-security-guidance.local.md`	Surcharges locales (à mettre dans `.gitignore`).

Exemple de règle : « Tout appel `requests.get(url)` avec une URL fournie par l'utilisateur passe par `acme.net.safe_request` ». Budget total de 8 Ko ; seule la couche 2 lit ce fichier.

Conseils d'utilisation

- **Faire taire un faux positif** : ajouter sur la ligne un commentaire qui explique pourquoi c'est sûr. Pour une exclusion générale, l'écrire dans `claude-security-guidance.md`.
- **Trop de faux positifs** : essayer un modèle moins cher et comparer ; pour la précision, rester sur Opus.
- **Rien ne se déclenche** : lancer `/hooks` dans une session ouverte à la racine du dépôt pour voir les hooks enregistrés, puis lire `~/.claude/security/log.txt`.
- **Aucun secret dans le fichier de règles** : il est envoyé au modèle à chaque revue.
- **Ne remplace pas une vraie revue de sécurité** (SAST/DAST, audit des dépendances, pentest) : c'est un filet de sécurité, sans garantie.

Sources (documentation officielle)

- README du plugin
- Documentation – Catch security issues as Claude writes code
- A Practical Guide to Claude Code's Security-Guidance Plugin – Medium