wycheproof

wycheproof

> *Catalogue généré le 2026-05-11*

En une phrase

Wycheproof est une énorme collection de "pièges crypto" maintenus par Google : des entrées tordues, déjà connues pour casser les implémentations bancales, qu'on jette à ta lib crypto pour voir si elle tient le coup.

Quand l'utiliser

- Tester une implémentation maison ou tierce d'un algo crypto (RSA, AES, ECDSA, HMAC…).
- Auditer une bibliothèque crypto avant intégration dans un produit.
- Vérifier qu'une lib supporte correctement les courbes elliptiques (curves).
- Tester un wallet ou un client blockchain qui valide des signatures.
- Vérifier que ta lib rejette bien les cas vicieux (zéro, courbes invalides, paddings cassés…).

Comment l'invoquer

- **Slash command** : `/wycheproof`
- **Phrases déclencheurs (texte)** : "crypto test vectors", "Wycheproof tests", "cryptographic edge cases"
- **Auto-invocation** : Sur demande explicite

Description détaillée

Wycheproof, c'est un trésor pour qui audite de la crypto. C'est une collection massive de "test vectors" — des couples entrée/sortie connus qui correspondent à des cas d'attaque réels documentés au fil des années. Originalement développé par Google, c'est maintenant un projet communautaire où chercheurs et industriels contribuent leurs trouvailles.

Pourquoi c'est précieux ? Implémenter de la crypto correctement, c'est terriblement dur. Un détail oublié (un padding mal géré, un point sur une courbe invalide accepté, un IV réutilisé…) suffit à briser toute la sécurité. Wycheproof contient des entrées qui ressemblent à de la crypto valide mais qui exploitent des bugs subtils. Si ta lib accepte une signature falsifiée, déchiffre quelque chose qu'elle devrait refuser, ou plante sur un input border-line — Wycheproof le révèle.

Concrètement, tu télécharges les fichiers JSON de test vectors correspondant à ton algo (par exemple "ECDSA secp256k1"), tu écris un petit harness qui boucle dessus et applique ta lib, et tu compares le résultat attendu (`valid`, `invalid`, `acceptable`) avec ce que ta lib renvoie. Tout écart = bug potentiel. C'est devenu un standard de validation crypto, notamment dans l'écosystème blockchain où les bugs de validation de signature peuvent coûter des millions.

Pour aller plus loin

Pour les exemples concrets, options de configuration et patterns avancés, voir le SKILL.md original.

Source

- **Plugin** : `trailofbits/testing-handbook-skills`
- **Nom interne** : `wycheproof`
- **Fichier** : `/home/thymon/.claude/plugins/cache/trailofbits/testing-handbook-skills/1.0.1/skills/wycheproof/SKILL.md`

Revision #2
Created 2026-05-11 21:19:47 UTC by thymon
Updated 2026-05-11 21:37:22 UTC by thymon