token-integration-analyzer

token-integration-analyzer

> *Catalogue généré le 2026-05-11*

En une phrase

Analyse soit le contrat d'un token (ERC20/ERC721) pour vérifier sa conformité, soit un protocole qui intègre des tokens externes pour vérifier qu'il gère bien les ~24 "tokens bizarres" qui existent (USDT sans return value, fee-on-transfer, rebasing, etc.).

Quand l'utiliser

- Quand tu construis un token (ERC20 ou NFT ERC721) et que tu veux t'assurer qu'il respecte le standard.
- Quand tu construis un protocole DeFi qui **accepte des tokens arbitraires** (un DEX, un lending market, un yield aggregator) — c'est là que ça devient critique : un token bizarre peut casser ta logique.
- Pour analyser les "weird ERC20 patterns" : USDT qui ne renvoie pas de boolean sur `transfer`, tokens à frais (PAXG), tokens rebasing (Ampleforth), tokens pausables (BNB), tokens upgradeables (USDC), tokens avec hooks (ERC777 → réentrance), etc.
- Pour faire une analyse on-chain : distribution des holders, supply, présence sur exchanges, risques de flash mint.
- Quand tu hésites sur la nécessité d'utiliser `SafeERC20` ou un wrapper défensif.

Comment l'invoquer

- **Slash command** : `/token-integration-analyzer`

- **Phrases déclencheurs (texte)** : "analyze token integration" / "check ERC20 conformity" / "audit weird token handling"
- **Auto-invocation** : Sur demande explicite.

Description détaillée

Le grand piège du DeFi : tous les "ERC20" ne se comportent pas comme le standard le dit. USDT ne retourne pas de booléen sur `transfer` (les protocoles naïfs cassent). PAXG prend des frais à chaque transfert (tu envoies 100, le destinataire reçoit 98). Ampleforth modifie les balances sans émettre d'event. ERC777 a des hooks qui permettent la réentrance. Si ton protocole ne sait pas gérer ces cas, il perd des fonds.

Ce skill, basé sur la "Token Integration Checklist" + "Weird ERC20 Database" de Trail of Bits, fait deux choses selon le contexte. **Si tu écris un token** : il vérifie la conformité ERC20/ERC721 (return values, decimals, métadonnées, race conditions sur approve), liste les privilèges du owner (mint, pause, blacklist), analyse la complexité du contrat. **Si ton protocole intègre des tokens externes** : il check les 10 catégories d'assessment dont les fameux ~24 "weird patterns" — fee-on-transfer, rebasing, missing return values, hooks de réentrance, balances modifiées hors transfer, low/high decimals, code injection via le nom du token, etc.

Pour Solidity, il s'appuie sur `slither-check-erc`, `slither --print human-summary` et `slither-prop`. Pour les contrats déjà déployés, il peut aussi faire de l'analyse on-chain (supply, distribution, exchange listings).

Pour aller plus loin

Pour la liste complète des 24+ weird ERC20 patterns avec exemples et mitigations, consulter le SKILL.md original à `/home/thymon/.claude/plugins/cache/trailofbits/building-secure-contracts/1.0.1/skills/token-integration-analyzer/SKILL.md` et la ressource `resources/ASSESSMENT_CATEGORIES.md`.

Source

- **Plugin** : `trailofbits/building-secure-contracts`
- **Nom interne** : `token-integration-analyzer`
- **Fichier** : `/home/thymon/.claude/plugins/cache/trailofbits/building-secure-contracts/1.0.1/skills/token-integration-analyzer/SKILL.md`

Revision #2
Created 2026-05-11 21:19:33 UTC by thymon
Updated 2026-05-11 21:37:10 UTC by thymon