algorand-vulnerability-scanner

algorand-vulnerability-scanner

> *Catalogue généré le 2026-05-11*

En une phrase

Scanne les smart contracts Algorand (écrits en TEAL ou PyTeal) pour repérer 11 types de failles classiques sur cette chaîne, dont la plus emblématique : l'attaque par "rekeying" (donner l'autorité d'un compte à un attaquant).

Quand l'utiliser

- Quand tu auditeras un projet Algorand (DeFi, NFT, application stateful).
- Avant de pousser une "logic signature" (smart contract sans état) en prod.
- Pour vérifier qu'un programme TEAL bloque bien la "rekeying attack" (= modifier secrètement la clé d'autorité d'un compte pour en prendre le contrôle).
- Quand le projet utilise des transactions groupées atomiques ("atomic groups") — l'ordre des transactions peut être manipulé.
- Pour tester rapidement avec **Tealer** (l'analyseur statique Trail of Bits dédié à Algorand).

Comment l'invoquer

- **Slash command** : `/algorand-vulnerability-scanner`
- **Phrases déclencheurs (texte)** : "audit my Algorand contract" / "check PyTeal contract" / "scan TEAL approval program"
- **Auto-invocation** : Sur demande explicite (généralement lors d'un audit).

Description détaillée

Algorand est une blockchain "pure proof-of-stake" avec un modèle de transaction très particulier : chaque transaction porte beaucoup de champs (RekeyTo, CloseRemainderTo, AssetCloseTo...) que le contrat doit vérifier explicitement. Si tu oublies de vérifier `RekeyTo`, un attaquant peut détourner ton compte. Les contrats sont écrits en TEAL (assembleur) ou plus souvent en PyTeal (DSL Python qui compile en TEAL).

Ce skill couvre 11 patterns : 1) **Rekeying Vulnerability** (CRITICAL — oubli du check RekeyTo), 2) **Missing Transaction Verification** (pas de check GroupSize/GroupIndex), 3) **Group Transaction Manipulation**, 4) **Asset Clawback Risk**, 5) **Application State Manipulation**, 6) **Asset Opt-In Missing**, 7) **Minimum Balance Violation**, 8) **Close Remainder To Check** (oubli qui peut vider un compte), 9) **Application Clear State** (peut être abusé pour bypass), 10) **Atomic Transaction Ordering** (supposer un ordre sans le valider), 11) **Logic Signature Reuse** (sigs réutilisables = replay attack).

Le skill cherche les fichiers `.teal` et `.py` (PyTeal), peut lancer **Tealer** (`pip3 install tealer; tealer contract.teal --detect all`) et te livre un rapport hiérarchisé.

Pour aller plus loin

Pour les détails techniques (patterns TEAL/PyTeal complets, exemples de vulnérabilités exploitables, intégration Tealer et Beaker), consulter le SKILL.md original à `/home/thymon/.claude/plugins/cache/trailofbits/building-secure-contracts/1.0.1/skills/algorand-vulnerability-scanner/SKILL.md` et la ressource `resources/VULNERABILITY_PATTERNS.md`.

Source

- **Plugin** : `trailofbits/building-secure-contracts`
- **Nom interne** : `algorand-vulnerability-scanner`
- **Fichier** : `/home/thymon/.claude/plugins/cache/trailofbits/building-secure-contracts/1.0.1/skills/algorand-vulnerability-scanner/SKILL.md`

Revision #2
Created 2026-05-11 21:19:39 UTC by thymon
Updated 2026-05-11 21:37:16 UTC by thymon