cairo-vulnerability-scanner

cairo-vulnerability-scanner

> *Catalogue généré le 2026-05-11*

En une phrase

Scanne les smart contracts Cairo (le langage des dApps StarkNet, le "L2" de validité sur Ethereum) pour repérer 6 types de failles critiques liées à l'arithmétique felt252, au stockage, au contrôle d'accès et aux ponts L1/L2.

Quand l'utiliser

- Quand tu auditeras un contrat StarkNet écrit en Cairo (le langage spécifique de cet écosystème).
- Avant de mettre en prod un projet sur StarkNet (DeFi, NFT, etc.).
- Quand le projet inclut un pont L1-L2 (un mécanisme qui fait transiter des fonds entre Ethereum et StarkNet) — c'est souvent là que se trouvent les plus gros bugs.
- Pour valider les fonctions qui acceptent des messages venant d'Ethereum (`#[l1_handler]`).
- Si tu veux passer un coup de "Caracal" (l'analyseur statique Trail of Bits pour Cairo) avant l'audit externe.

Comment l'invoquer

- **Slash command** : `/cairo-vulnerability-scanner`
- **Phrases déclencheurs (texte)** : "audit my StarkNet contract" / "check Cairo contract" / "scan L1-L2 bridge"

- **Auto-invocation** : Sur demande explicite (généralement lors d'un audit).

Description détaillée

Cairo est le langage des smart contracts sur StarkNet, un "rollup de validité" qui exécute du code hors-Ethereum puis prouve cryptographiquement le résultat. Cairo a un type natif bizarre, le `felt252` (un entier sur 252 bits, presque comme un `uint256` mais pas tout à fait), qui crée des bugs d'arithmétique inattendus. Cairo 1.0 a changé pas mal de choses — beaucoup de codes existants ont des reliquats Cairo 0 dangereux.

Ce skill cherche 6 patterns critiques : 1) **Unchecked Arithmetic** (overflow/underflow sur `felt252`), 2) **Storage Collision** (deux variables qui se mappent au même slot de stockage — l'une écrase l'autre), 3) **Missing Access Control** (pas de vérification du caller sur des fonctions sensibles), 4) **Improper Felt252 Boundaries** (oublier que `felt252` n'est pas exactement un uint256), 5) **Unvalidated Contract Address** (utiliser une adresse fournie par l'utilisateur sans la valider), 6) **Missing Caller Validation** (oublier `get_caller_address()` sur une fonction admin).

Il peut aussi exécuter **Caracal**, l'outil officiel Trail of Bits (`caracal detect src/`), pour compléter avec des règles automatisées.

Pour aller plus loin

Pour les détails techniques (exemples Cairo 1.0, patterns L1 handlers, intégration Caracal et Starknet Foundry), consulter le SKILL.md original à `/home/thymon/.claude/plugins/cache/trailofbits/building-secure-contracts/1.0.1/skills/cairo-vulnerability-scanner/SKILL.md` et la ressource `resources/VULNERABILITY_PATTERNS.md`.

Source

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

Revision #2
Created 2026-05-11 21:19:38 UTC by thymon
Updated 2026-05-11 21:37:15 UTC by thymon