ton-vulnerability-scanner

ton-vulnerability-scanner

> *Catalogue généré le 2026-05-11*

En une phrase

Scanne les smart contracts TON (The Open Network, la blockchain de Telegram) écrits en FunC pour repérer 3 types de failles critiques liées à la validation de l'expéditeur, aux dépassements arithmétiques et à la gestion du gas.

Quand l'utiliser

- Quand tu auditeras un contrat TON ou Jetton (le standard de token sur TON).
- Avant de déployer un contrat FunC en prod sur le mainnet TON.
- Quand le projet implique un wallet TON ou un système de notifications de transfert (un piège classique : croire qu'un message vient du vrai contrat Jetton alors qu'il vient d'un faux).
- Pour vérifier que les opérations privilégiées (mint, burn, withdraw) valident bien le sender.
- Quand tu transfères du TON entre contrats et que tu veux t'assurer que le gas est correctement réservé pour éviter que la transaction échoue à mi-chemin.

Comment l'invoquer

- **Slash command** : `/ton-vulnerability-scanner`
- **Phrases déclencheurs (texte)** : "audit my TON contract" / "check FunC contract" / "scan Jetton implementation"
- **Auto-invocation** : Sur demande explicite (généralement lors d'un audit).

Description détaillée

TON est la blockchain associée à Telegram, avec une architecture très différente d'Ethereum (modèle asynchrone à base de messages). Les contrats sont en FunC, un langage bas niveau. Les bugs TON les plus courants sont liés à la nature "message-passing" : un contrat reçoit un message et doit vérifier qui l'envoie, sinon n'importe qui peut déclencher des actions privilégiées.

Ce skill cherche 3 patterns critiques : 1) **Missing Sender Check** (le contrat ne vérifie pas l'adresse de l'expéditeur — quelqu'un peut appeler une fonction admin), 2) **Integer Overflow** (l'arithmétique FunC déborde silencieusement si non vérifiée — perte ou création de tokens), 3) **Improper Gas Handling** (mauvaise réservation de gas lors d'un `send_raw_message` — la transaction échoue à mi-parcours et laisse des états incohérents).

Il regarde aussi les pièges Jetton : un faux contrat Jetton peut envoyer un message `transfer_notification` pour piéger un contrat naïf. Le workflow : tu pointes le dossier `contracts/` avec tes `.fc`, le skill analyse et te liste les findings avec recommandations.

Pour aller plus loin

Pour les détails techniques (patterns FunC, exemples Jetton, recommandations de gas), consulter le SKILL.md original à `/home/thymon/.claude/plugins/cache/trailofbits/building-secure-contracts/1.0.1/skills/ton-vulnerability-scanner/SKILL.md` et la ressource `resources/VULNERABILITY_PATTERNS.md`.

Source

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

Revision #2
Created 2026-05-11 21:19:38 UTC by thymon
Updated 2026-05-11 21:37:14 UTC by thymon