address-sanitizer

address-sanitizer

> *Catalogue généré le 2026-05-11*

En une phrase

AddressSanitizer (ASan) est un "détecteur de fumée" qu'on greffe au code C/C++ pour repérer en temps réel les bugs mémoire (buffer overflow, use-after-free) que l'œil ne voit jamais.

Quand l'utiliser

- Pendant une campagne de fuzzing C/C++ (combo classique fuzzer + ASan).
- Pour tester du code Rust qui utilise `unsafe`.
- Pour traquer un bug mémoire suspect (corruption silencieuse, plantage aléatoire).
- Pour auditer une bibliothèque qui manipule beaucoup de pointeurs.
- Avant un release important d'un projet C ou C++.

Comment l'invoquer

- **Slash command** : `/address-sanitizer`
- **Phrases déclencheurs (texte)** : "ASan", "memory error detection", "AddressSanitizer setup"
- **Auto-invocation** : Sur demande explicite

Description détaillée

AddressSanitizer (ASan pour les intimes) est un outil de "sanitisation" : il instrumente ton code à la compilation pour ajouter des vérifications partout où ton programme touche à la mémoire. Du coup, dès qu'il détecte un accès illégal (lire un octet hors d'un tableau, utiliser un pointeur déjà libéré, double free…), il arrête le programme et te dit exactement où ça s'est passé, avec une stack trace propre.

C'est devenu le standard de l'industrie pour le fuzzing C/C++. Pourquoi ? Parce que beaucoup de bugs mémoire ne font pas planter le programme immédiatement — ils corrompent silencieusement la mémoire et causent des bugs aléatoires plus tard. Sans ASan, ton fuzzer pourrait passer à côté. Avec ASan, chaque accès interdit est attrapé sur le fait.

Le coût : ASan ralentit l'exécution de 2 à 4 fois et réserve 20TB de mémoire virtuelle (oui, terabytes — mais c'est virtuel, pas vraiment consommé). Il faut compiler avec `-fsanitize=address`. ASan détecte les classiques : buffer overflow (lecture/écriture hors zone), use-after-free (utilisation d'un pointeur libéré), double-free, memory leaks. À combiner avec UndefinedBehaviorSanitizer pour une couverture maximale.

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** : `address-sanitizer`
- **Fichier** : `/home/thymon/.claude/plugins/cache/trailofbits/testing-handbook-skills/1.0.1/skills/address-sanitizer/SKILL.md`

Revision #2
Created 2026-05-11 21:19:54 UTC by thymon
Updated 2026-05-11 21:37:28 UTC by thymon