coverage-analysis

coverage-analysis

> *Catalogue généré le 2026-05-11*

En une phrase

L'analyse de couverture mesure quelles lignes de ton code ont vraiment été exécutées par les tests/fuzzers — pour repérer les "zones aveugles" que personne ne teste.

Quand l'utiliser

- Évaluer si une campagne de fuzzing visite vraiment le code intéressant (ou bloque ailleurs).
- Identifier les parties d'un programme jamais touchées par les tests.
- Comparer deux versions d'un harness pour voir laquelle explore plus de code.
- Détecter les "fuzzing blockers" (un checksum, une validation qui bloque tout).
- Prioriser les améliorations de tests (sur les zones non couvertes).

Comment l'invoquer

- **Slash command** : `/coverage-analysis`
- **Phrases déclencheurs (texte)** : "code coverage", "fuzzing coverage", "harness effectiveness"
- **Auto-invocation** : Sur demande explicite

Description détaillée

La couverture de code, c'est l'art de répondre à la question : "quelles lignes/branches de mon code sont vraiment exécutées quand je lance mes tests ?". On instrumente le code à la compilation (avec `-fprofile-instr-generate` par exemple), on le lance avec ses tests/fuzzers, et on en sort un rapport qui te montre en vert les lignes visitées et en rouge celles jamais touchées.

En fuzzing, c'est doublement utile. D'abord pour évaluer ton harness : si après plusieurs heures de fuzzing tu n'as couvert que 30 % du code de ta cible, c'est probablement que ton harness est mal pensé ou qu'un obstacle (checksum, validation) bloque tout. Ensuite pour suivre les progrès : tu modifies le harness, tu rajoutes un dictionnaire, tu changes de fuzzer, et la couverture te dit immédiatement si ça aide ou pas.

La couverture n'est pas un indicateur parfait de la qualité du fuzzing (un mauvais fuzzer peut atteindre une bonne couverture sans trouver de bugs), mais c'est le meilleur indicateur disponible et facile à mesurer. Les outils classiques : `llvm-cov` (pour le code compilé avec Clang), `gcov` (pour GCC), `lcov` pour générer des rapports HTML, et des plateformes comme Codecov pour suivre la couverture dans le temps. À utiliser systématiquement quand on prend le fuzzing au sérieux.

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

Revision #2
Created 2026-05-11 21:19:44 UTC by thymon
Updated 2026-05-11 21:37:20 UTC by thymon