aflpp

aflpp

> *Catalogue généré le 2026-05-11*

En une phrase

AFL++ est un "fuzzer" (machine à tester) qui bombarde un programme C/C++ avec des milliers d'entrées aléatoires en parallèle sur plusieurs cœurs CPU pour faire sortir tous les bugs et plantages cachés.

Quand l'utiliser

- Tester intensivement un projet écrit en C ou C++ pour traquer les bugs mémoire.
- Faire tourner une grosse campagne de fuzzing sur plusieurs cœurs (très rapide).
- Pousser un test plus loin quand un fuzzer plus simple (libFuzzer) a déjà épuisé ses idées.
- Auditer un logiciel mature avant un release important.
- Auditer une bibliothèque utilisée par beaucoup de monde (sécurité critique).

Comment l'invoquer

- **Slash command** : `/aflpp`
- **Phrases déclencheurs (texte)** : "fuzz this C/C++ project", "multi-core fuzzing", "AFL++"
- **Auto-invocation** : Sur demande explicite

Description détaillée

AFL++ (American Fuzzy Lop, version "plus plus") est un fuzzer de référence pour le code C et C++. Un fuzzer, c'est un outil qui prend ta fonction et lui balance des millions d'entrées bizarres et aléatoires jusqu'à ce qu'elle plante ou se comporte mal. C'est un peu comme tester une porte en lui donnant des coups dans tous les sens pour voir si elle craque.

Son gros atout par rapport à ses concurrents : il sait fuzzer sur plusieurs cœurs CPU en parallèle de façon stable. Du coup, il est idéal pour des grosses campagnes (laisser tourner une nuit ou un week-end sur 8 cœurs). Il est plus complexe à installer que libFuzzer (il faut Clang ou GCC, parfois Docker), mais il offre des "mutations" plus variées et un mode CMPLOG qui aide à franchir les vérifications difficiles dans le code.

Quand tu auditerais ce qu'il a trouvé, AFL++ te livre des "crashes" (entrées qui ont fait planter), des "hangs" (entrées qui ont fait freezer) et des stats de couverture (quelles parties du code ont été visitées). On le combine souvent avec AddressSanitizer pour détecter les corruptions mémoire silencieuses.

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

Revision #2
Created 2026-05-11 21:19:43 UTC by thymon
Updated 2026-05-11 21:37:19 UTC by thymon