Analyse de vulnerabilités : Vers une exécution symbolique plus efficiente

Frédéric Recoules and Sébastien Bardin

CEA, List

frederic.recoules@cea.fr sebastien.bardin@cea.fr
https://binsec.github.io/

Les méthodes d'analyse de programme de type exécution symbolique [2] sont reconnues comme prometteuses pour la découverte automatisée de bugs et de vulnérabilités. Ces méthodes se basent notamment sur des raisonnements logiques au niveau du programme pour identifier automatiquement des entrées menant à des erreurs.

La plateforme d'analyse de code binaire BINSEC [4, 1], développée au CEA utilise différentes variantes d'exécutions symboliques [3, 6, 5]. Le passage à l'échelle de la technique est essentiel pour mener à bien les analyses. Ce stage aura pour objectif de contribuer à l'amélioration au sens large de la plateforme et de son évaluation contre les outils concurrents sur des benchmarks de l'état de l'art. Selon les goûts du stagiaire, les tâches proposées peuvent être axées algorithmique de performance (ex : meilleure exploration de l'espace de recherche, amélioration des raisonnements logiques), fonctionalités (ex : ajouter de nouveaux moniteurs de bugs), expérimentations ou robustification du code. Une bonne maitrise du langage OCaml est attendue.

Éléments logistiques. Le stage sera hébergé sur le site Nano-INNOV à Saclay. Le stage sera co-encadré par Frédéric Recoules et Sébastien Bardin.

Postuler. Pour candidater ou pour obtenir plus d'information, merci de prendre contact par mail avec les encadrants.

Références

[1] Binsec. https://binsec.github.io/.

[2] Cristian Cadar and Koushik Sen. Symbolic execution for software testing : Three decades later. *Commun. ACM*, 2013.

[3] Lesly-Ann Daniel, Sébastien Bardin, and Tamara Rezk. Binsec/rel : Symbolic binary analyzer for security with applications to constant-time and secret-erasure. *ACM Trans. Priv. Secur.*, 2023.

[4] Adel Djoudi and Sébastien Bardin. Binsec : Binary code analysis with low-level regions. In *Tools and Algorithms for the Construction and Analysis of Systems*, 2015.

[5] Soline Ducousso, Sébastien Bardin, and Marie-Laure Potet. Adversarial reachability for program-level security analysis. In Thomas Wies, editor, *Programming Languages and Systems*, 2023.

[6] Guillaume Girol, Benjamin Farinier, and Sébastien Bardin. Not all bugs are created equal, but robust reachability can tell the difference. In Alexandra Silva and K. Rustan M. Leino, editors, *Computer Aided Verification*, pages 669–693, Cham, 2021. Springer International Publishing.