Extension d'une analyse de vulnérabilité de code binaire à des attaques par injection de fautes

Yanis Sellami
CEA, List

Sébastien Bardin
CEA, List

yanis.sellami@cea.fr, sebastien.bardin@cea.fr

L'analyse automatique de programmes est une discipline qui vise à construire des analyses et des outils pour détecter des bugs dans le code des programmes. Les techniques développées permettent notamment de détecter les vulnérabilités mémoire (*e.g. buffer overflows*) ou numériques (*e.g.* division par zéro) qui peuvent corrompre les données ou stopper l'exécution. Pourtant, même lorsqu'on peut prouver qu'un programme est exempt de tels bugs, un attaquant peut exploiter des éléments physiques du matériel sur lequel le code est exécuté pour modifier le code où les données pendant l'exécution. L'attaquant peut alors contourner des mesures de sécurité présentes dans le code.

La recherche automatique de vulnérabilités aux attaques par injection de fautes est cependant coûteuse car chaque faute augmente la surface du code à analyser. La plateforme d'analyse de code binaire BINSEC [1], développée au CEA, propose une méthode à l'état de l'art pour détecter des attaques avec fautes [2]. Toutefois, cette technique possède encore plusieurs limitations (notamment, pas de gestion des fautes permanentes) ce qui l'empêche de détecter certaines attaques de l'état de l'art (*e.g.* rowhammer [3]). On pourra également considérer des améliorations ou optimisations de la technique d'analyse afin d'en améliorer les performances et de pouvoir détecter des vulnérabilités nécessitant plus de fautes ou sur des codes plus larges.

L'objectif de ce stage est de proposer une implémentation d'extension ou d'optimisation de la technique d'analyse en fautes de BINSEC.

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

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

Références

[1] "Plateforme binsec." `https://binsec.github.io/`.

[2] S. Ducousso, S. Bardin, and M.-L. Potet, "Adversarial reachability for program-level security analysis," in *Programming Languages and Systems* (T. Wies, ed.), (Cham), pp. 59–89, Springer Nature Switzerland, 2023.

[3] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them : an experimental study of dram disturbance errors," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, p. 361–372, IEEE Press, 2014.