Vers une exécution symbolique plus robuste :
Modélisation de fonctions de librairies et appels système

Frédéric Recoules et 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 [3, 2] sont reconnues comme prometteuses pour la découverte automatisée de bugs et de vulnérabilités. La plateforme d'analyse de code binaire BINSEC [5, 1], développée au CEA propose par exemple différentes variantes d'exécutions symboliques [4, 7, 6].

Fournir un mécanisme de modélisation de la sémantique de fragments de code (communément appelé "*stubbing*") est essentiel pour la complétude et le passage à l'échelle de l'analyse. Un *stub* peut permettre de combler l'absence de code (fonctions externes, appels système) ou de simplifier l'implémentation réelle d'une partie du programme. Les fonctions de librairies sont généralement de bonnes cibles pour réintroduire des concepts de plus haut-niveau (`string` [8], système de fichiers, etc.).

Ce stage aura pour objectif de faire l'état des lieux des facultés existantes et des besoins de *stubbing* de la plateforme BINSEC. Plusieurs pistes pourront ensuite être explorées selon les préférences du stagiaire, telles que la généralisation et la robustification des *stubs* existants (ex : indépendance à l'architecture, mise en commun avec d'autres outils), l'intégration de concepts avancés dans BINSEC (ex : système de fichiers) ou encore la recherche de primitives facilitant l'expression de la sémantique d'*objets* couramment manipulés (ex : `string`, `vector`). Une bonne maîtrise 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] Plateforme binsec. `https://binsec.github.io/`.

[2] Roberto Baldoni, Emilio Coppa, Daniele Cono D'Elia, Camil Demetrescu, and Irene Finocchi. A survey of symbolic execution techniques. *ACM Comput. Surv.*, 2018.

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

[4] 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.

[5] 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.

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

[7] Guillaume Girol, Benjamin Farinier, and Sébastien Bardin. Not all bugs are created equal, but robust reachability can tell the difference. In *Computer Aided Verification*, 2021.

[8] Frederico Ramos, Nuno Sabino, Pedro Adão, David A. Naumann, and José Fragoso Santos. Toward Tool-Independent Summaries for Symbolic Execution. In *37th European Conference on Object-Oriented Programming (ECOOP 2023)*, 2023.