Le projet LEIA : Learning-Enhanced Incremental Analyses

Virgile Prevosto

Université Paris-Saclay, CEA, List

1 Description

Le projet LEIA s'inscrit dans le cadre du Grand Défi Cyber-sécurité, et regroupe 3 équipes du CEA List ainsi que la société Systerel. Le projet se base sur des outils matures (TRL 7-8), mais se propose de leur apporter des extensions beaucoup plus exploratoires (TRL 3-4). Le projet a démarré en juin 2021 pour une durée d'un an, éventuellement suivie d'une participation à la deuxième phase du grand défi.

L'objectif principal du projet est de faciliter l'intégration d'outils d'analyse formelle de code, en particulier Frama-C (`https://frama-c.com`) et Binsec (`https://binsec.github.io`) dans des cycles de développement plus agiles que le traditionnel cycle en V des systèmes critiques, pour vérifier des propriétés de sécurité.

Dans ce contexte, quatre axes de travaux principaux ont été définis. Tout d'abord, le langage de spécification ACSL utilisé pour spécifier des propriétés à vérifier sur des programmes analysés par Frama-C n'est pas toujours bien adapté pour exprimer des propriétés de sécurité. S'il reste théoriquement possible de le faire, c'est souvent au prix d'un encodage lourd, peu lisible, et sujet à erreurs s'il est fait manuellement. De ce fait, il est important de fournir un langage de plus haut niveau qu'ACSL dédié à l'expression de ces propriétés.

Dans le même ordre d'idée, écrire des spécifications formelles, et s'assurer qu'elles correspondent bien aux exigences décrites en langage naturel, requiert une expertise certaine et assez peu répandue. LEIA se propose donc de s'appuyer sur des outils basés sur le Traitement Automatique des Langues (TAL) pour extraire les concepts les plus importants des exigences, voire proposer directement des annotations formelles.

Par ailleurs, les outils d'analyse de code et les solveurs associés demandent souvent un paramétrage très fin pour optimiser leurs résultats en fonction du temps de calcul nécessaire pour les obtenir. Ce paramétrage est à l'heure actuel le plus souvent manuel et demande du temps. LEIA va utiliser des algorithmes d'apprentissage pour proposer automatiquement des jeux de paramètres adéquats en fonction des caractéristiques du code sous analyse ou des formules

logiques générées, permettant là encore de rendre la prise en main des outils beaucoup plus simple.

Enfin, les analyseurs n'ont généralement pas de notion d'évolution du code : si on soumet une nouvelle version d'un programme, elle sera traitée comme une nouvelle base de code. Comme les temps d'analyse pour une vérification formelle sont généralement non négligeables, ce point peut vite devenir bloquant s'il y a peu de différences entre les deux versions, par exemple dans un contexte d'intégration continue. LEIA va donc étendre ses outils pour qu'ils puissent prendre en compte les correspondances entre ces versions et réutiliser autant que possible les résultats d'analyses précédentes.

Les différents développements effectués seront en outre déployés sur des études de cas, notamment la bibliothèque sécurisée S2OPC (`https://gitlab.com/systerel/S2OPC/` implémentant le protocole de communication OPC-UA.

2 Principaux résultats

Des premiers développements ont été réalisé dans tous les axes décrits précédemment. Tout d'abord, les annotations de haut niveau sont exprimables grâce au greffon MetAcsl (`https://git.frama-c.com/pub/meta`), notamment pour exprimer des propriétés de confidentialité et d'intégrité. Par ailleurs, l'outil Lima a permis de construire un moteur de recherche sémantique sur la norme OPC-UA, et une petite base d'exemples de correspondance anglais/ACSL a été constituée pour de premières expérimentations, même si elle reste trop modeste pour envisager d'obtenir une traduction automatique complète à ce stade.

Le troisième axe a connu différents développements. En premier lieu, des premières expériences ont eu lieu pour piloter les heuristiques du solveur S3 par des mécanismes d'apprentissage en fonction des caractéristiques des formules à vérifier. De même, un apprentissage a été effectué pour permettre au greffon Eva de Frama-C de décider automatiquement s'il doit ou non dérouler une boucle donnée lors de son analyse. Cet apprentissage est basé sur des critères purement syntaxiques, et une version plus sémantique, basée sur l'embedding de graphes est en cours d'implémentation. Enfin, les techniques de déobfuscation de code de Binsec ont été enrichis dans l'outil Xynthia (`https://github.com/binsec/xyntia`) par des S-métaheuristiques basées sur l'observation des entrées et sorties du programme.

Enfin, de premières expérimentations ont été réalisées dans Frama-C pour permettre à l'analyseur Eva de réutiliser des états abstraits calculés précédemment, et au noyau de fournir des informations sur les correspondances existant entre deux arbres de syntaxe abstraite du programme.