Analyse de la robustesse de code binaire

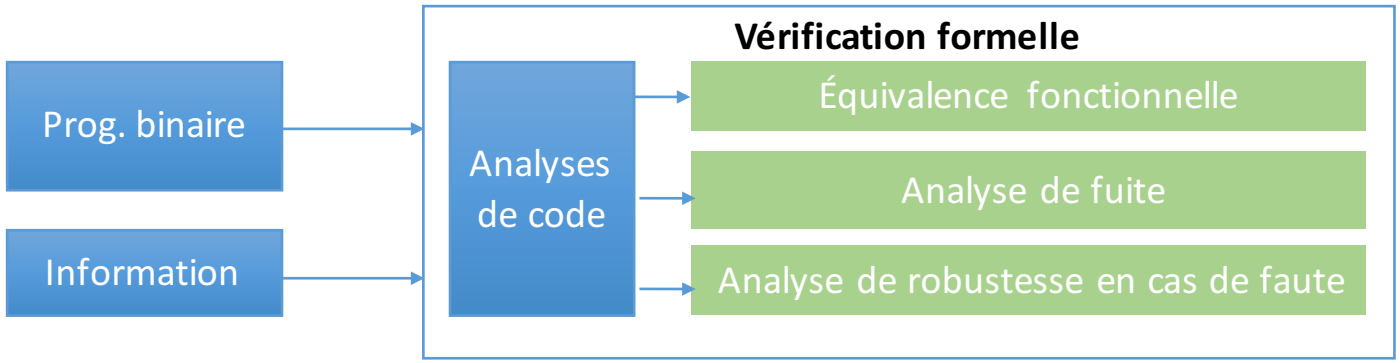
I. Ben El Ouahma, J-B Bréjon, E. Encrenaz,
K. Heydemann, Q. Meunier, S. T. Vu

LIP6

AGENCE NATIONALE DE LA RECHERCHE
ANR

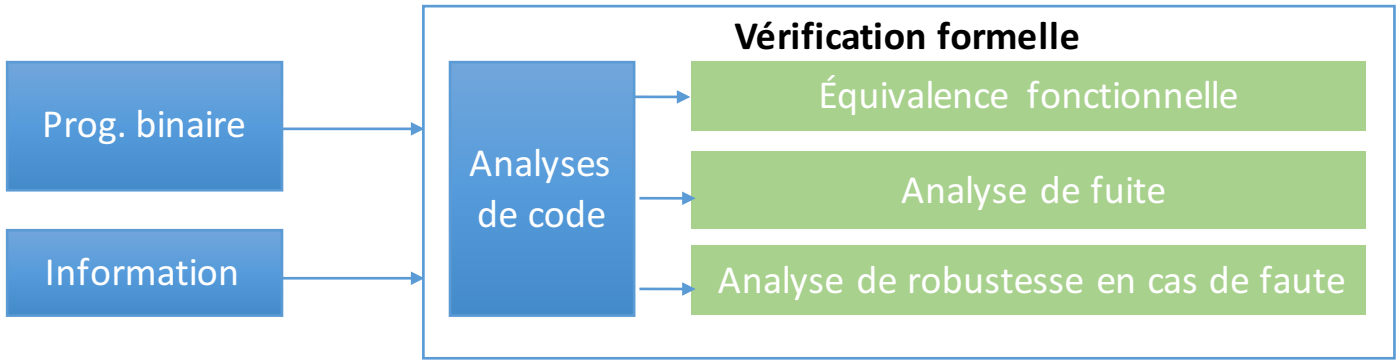
SORBONNE UNIVERSITÉ

Objectifs



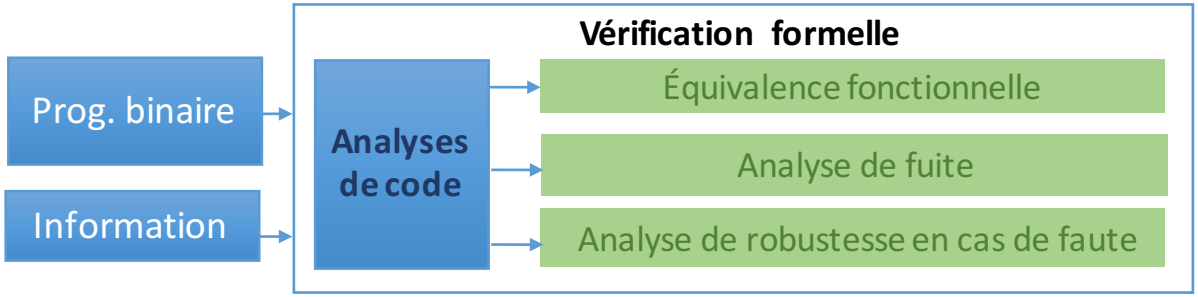
- Analyse de robustesse du code final nécessaire
  - Post-sécurisation, post-compilation
  - Code final :
    - instructions et leur agencement, placement du code,
    - registres alloués et implantation des données
- Objectif global : proposer une plate-forme d'outils permettant
  - de localiser des vulnérabilités / attester la robustesse de code binaire
  - d'analyser différents schémas de sécurisation
  - d'analyser les compatibilités ou incompatibilités entre différentes protections

État des travaux



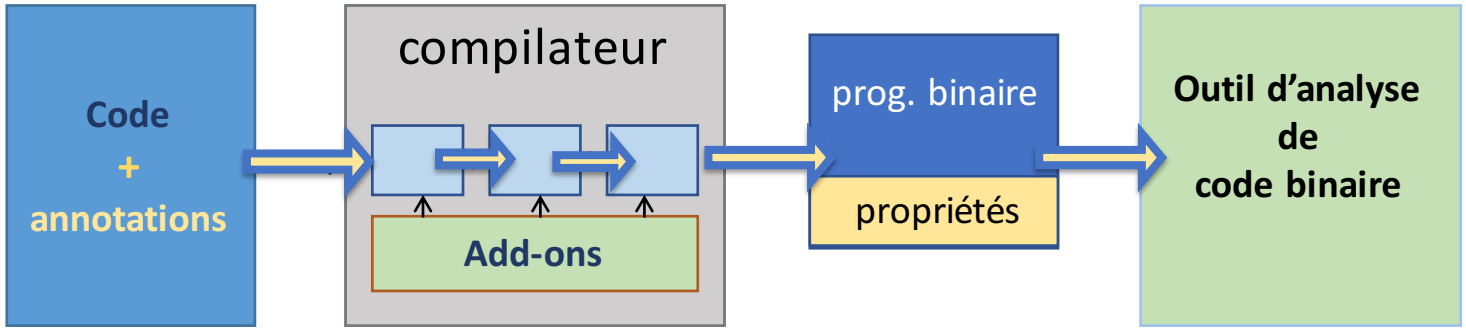
- Analyses définies
  - Equivalence fonctionnelle (définie mais non automatisée)
  - Analyse symbolique de fuite de code assembleur masqué (présentation de Q. Meunier)
  - Analyse de robustesse de code soumis à des fautes transitoires (présentation de J-B Bréjon)
- Automatisation
  - Binaires ARMv7-M supportés
  - Combinaison d'analyses statiques et symboliques en amont des techniques de vérification définies

Fonctionnement global



- Combinaison d'analyses statiques et symboliques
  - Analyse statique pour la récupération d'une représentation du code
    - CFG + instructions avec leur adresse et encodage
    - information sur les données globales (var. secrètes, masques, …)
  - Analyse symbolique pour récupération
    - « contexte d'exécution »
    - information sur les données locales (adresse Sbox masquée, …)
  - Basée sur `angr` (http://angr.io)
- Récupération de propriétés et informations produites par le compilateur
  - Thèse de Son Tuan Vu (oct. 2017 – sept 2020)
  - Annotation du code : du source au binaire
    - Guider les transformations et/ou exprimer une propriété utilisée ensuite par un outil d'analyse

Annotations de code : du source au binaire



- Langage d'annotations (code source)
  - Expression de propriétés sur code et/ou données
  - Basé sur un sous-ensemble *ANSI/ISO C Specification Language* (ACSL) + ajouts
- Propagation des annotations dans le compilateur (clang/LLVM)
  - Consommation et production, transformation, transport au travers des différentes passes
  - Difficultés liées aux optimisations
    - pipeline minimal avec quelques optimisations visé + warning si altération des informations détectée
- Production des informations dans le binaire au format DWARF