22888 MILED Youssef

Les méthodes formelles dans la ville connectée

L'utilisation de méthodes formelles permet de sécuriser les programmes informatiques, en s' assurant que ceux-ci respectent bien leurs spécifications. L'objectif est alors de tirer parti d'une de ces méthodes formelles pour garantir un certain niveau de sûreté, à savoir l'absence de transfert non voulu de donnés au sein du programme.

La vulnérabilité des objets connectés aux cyberattaques dans la ville connectée invite à chercher des méthodes efficaces pour remédier aux problèmes d'obtention illégale d'un accès au système ou de vol de données confidentielles, en sécurisant les programmes informatiques qui y interviennent.

Positionnement thématique (ÉTAPE 1) :

- INFORMATIQUE (Informatique Théorique)
- INFORMATIQUE (Informatique pratique)

Mots-clés (ÉTAPE 2)

Mots-clés (en français)	**Mots-clés (en anglais)**
Sécurité basée sur le langage	*Language-based security*
Confidentialité	*Confidentiality*
Intégrité	*Integrity*
Flux d'information	*Information flow*
Non-interférence	*Non-interference*

Bibliographie commentée

Les méthodes formelles jouent un rôle important pour sécuriser les systèmes informatiques. Au-delà du test, elles permettent d'assimiler un programme à une définition mathématique en utilisant une certaine sémantique, et donc utilisent la déduction logique pour établir les propriétés vraies pour toutes les entrées possibles du programme. [3]

On peut choisir de raisonner sur des modèles abstraits tels que les graphes de programmes (GP) qui sont des graphes représentant la structure des programmes étudiés. La vérification de programme (program verification) consiste à associer des prédicats à chaque nœud du GP, ce qui reste difficile à automatiser car les prédicats sont difficiles à exprimer en machine. Quant au model-checking, il permet de vérifier que le modèle d'abstraction du système satisfait au modèle formel des propriétés voulues grâce à des arbres de calcul logique qui expriment les propriétés de manière formelle tout en étant moins expressives que la première méthode afin d'obtenir une automatisation complète. Enfin, l'approche du language-based security (LBS) permet d'assurer la sécurité d'un programme dans un langage de programmation donné. [1]

En effet, cette approche se base notamment sur l'étude du flux d'information permis lors de l' exécution d'un programme. On distingue alors plusieurs niveaux de sécurité attribués à chaque entrée ou sortie intervenant dans le programme : cela se traduit en général par les ensembles {fiable, douteux} pour l'intégrité, {privé, public} ou aussi {non-classé, classé, secret, top secret} pour la confidentialité. [1]

Contrôler le flux d'information c'est dire si un programme peut laisser une suite d'informations d'une variable x dont le degré de confidentialité est faible vers une variable y dont le degré de confidentialité est élevé, par un flux direct (explicite) ou à cause d'une branche if-else ou autres (flux implicite). Le contrôle du flux d'information (Information Flow Control, IFC) a déjà été appliqué à certains langages de programmations comme Caml (Flow Caml). C'est ce que je me propose d'étudier, en me restreignant à une partie de Caml, et en créant un système de types de sorte à assurer qu'il n'y a pas de transfert non voulu de données. [2] [4]

Nevin Heintze et Jon G,Riecke conçoivent quant à eux le SLam calculus (Secure Lambda Calculus) qui est un lambda-calcul typé qui contient, en plus des types habituels, des informations sur la sécurité, à savoir lecteur, lecteur indirect (qui ont pour objectif de spécifier la confidentialité) ainsi que créateur et créateur indirect (qui ont pour objectif de spécifier l' intégrité). L'utilisation de langage de programmation pour la sécurité permet en ce sens de vérifier statiquement les programmes avant leur exécution [6]

Problématique retenue

En quoi l'approche de Language-based security nous permet d'édtudier la résolution des problèmes de confidentialité et d'intégrité dans les programmes informatiques ?

Objectifs du TIPE du candidat

Je me propose de :
- Définir formellement certains composants de la sécurité informatique à savoir confidentialité et intégrité
- Définir un système de type pour contrôler les flux d'information
- Prouver que le système de type est correct

Références bibliographiques (ÉTAPE 1)

[1] FLEMMING NIELSON, HANNE RIIS NIELSON : Formal methods, an appetizer *: https://link.springer.com*

[2] DAVID BASIN : Formal methods for security *: www.cybok.org*

[3] XAVIER LEROY : Le logiciel, entre l'esprit et la matière *: Leçon inaugurale prononcée au Collège de France le jeudi 15 novembre 2018 www.college-de-france.fr*

[4] VINCENT SIMONET : Flow Caml in a Nutshell *: http://cristal.inria.fr*

[5] NEVIN HEINTZE, JON G.RIECKE : The Slam Calculus : Programming with Secrecy and Integrity *: https://www.cs.cornell.edu/andru/cs711/2003fa/reading/heintze98slam.pdf*

Références bibliographiques (ÉTAPE 2)

[1] GEOFFREY SMITH : A New Type System for Secure Information Flow *: https://fpl.cs.depaul.edu/jriely/547/extras/smith-csfw01.pdf*

[2] DENNIS VOLPANO, GEOFFREY SMITH : A Type-Based Approach to Program Security *: https://www.researchgate.net/publication/220917257_A_Type-Based_Approach_to_Program_Security*

DOT

[1] *: Début septembre, recherches sur les méthodes formelles qui existent pour étudier mathématiquement les programmes informatiques.*

[2] *: En octobre, décision de travailler sur la sécurité des programmes informatiques en utilisant des méthodes formelles adéquates.*

[3] *: Fin février, choix du système de type (de Geoffrey Smith) à mettre en place pour répondre aux exigences de sécurité.*

[4] *: En mars, étude des propriétés des systèmes de type et compréhension des règles de typage.*

[5] *: Début mai, compréhension d'un théorème de non interférence et de sa preuve.*

[6] *: Mi-mai, mise en place d'une stratégie pratique pour vérifier, grâce au système de type, qu'un programme est conforme aux contraintes de confidentialité et d'intégrité, suivi d'une implémentation d'un algorithme qui vérifie qu'un arbre de dérivation de type est correct.*