RAVAJ

Réécriture pour la vérification d'applications Java

Quels sont les risques ?

Application honnête	Application malveillante

Voulez-vous envoyer un SMS ? Cancel Ok
`...`
`Connection.open("SMS:...")`
`...`
`...` *code de l'application*
	Voulez-vous jouer ? Cancel Ok
	`...`
	`Connection.open("SMS:...")`
	`...`
	`...` *code de l'application*

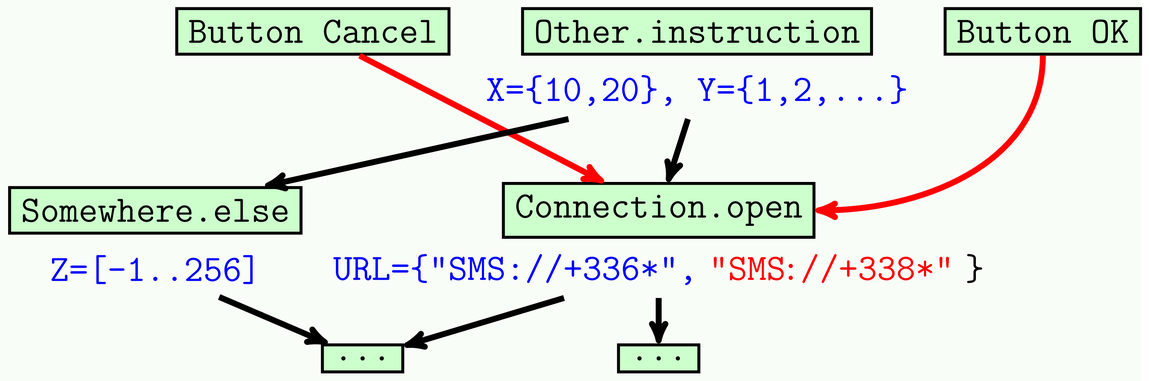
Dans cet exemple simple, l'application « honnête » demande la permission à l'utilisateur avant d'utiliser une ressource critique (ici l'envoi d'un SMS). A l'inverse, une application malveillante peut accéder aux ressources sans en informer l'utilisateur. Dans notre exemple, un SMS sera expédié sans que l'utilisateur en soit avisé.

Pour en savoir plus : Dans le cas d'applications Java MIDP pour téléphone mobile, les applications non certifiées par l'opérateur (la très grande majorité des applications disponibles) peuvent être exécutées dans un *mode sécurisé*. Dans ce mode, quelle que soit l'application, à *chaque fois* que celle-ci accède à une ressource critique, un message demande à l'utilisateur son accord pour l'utilisation de cette ressource. Ce procédé offre un très bon niveau de sécurité. En revanche, il n'est pas utilisable pour des applications utilisant *fréquemment* des ressources critiques (par exemple un navigateur internet), puisqu'il noie l'utilisateur sous des messages de confirmation. De telles applications doivent, préférablement, être certifiées par l'opérateur. Ceci a pour effet de désactiver les messages de sécurité. Cependant, l'opérateur doit avoir des garanties sur l'innocuité de l'application avant de la certifier. Ce qui revient au problème initial.

Comment éviter les mauvaises surprises ?

En vérifiant, à l'aide d'un *analyseur statique*, le code de l'application. Son rôle consiste en particulier à calculer une valeur approchée pour :

1. le **graphe d'appel** de l'application
2. l'ensemble des **valeurs possibles** des paramètres des méthodes



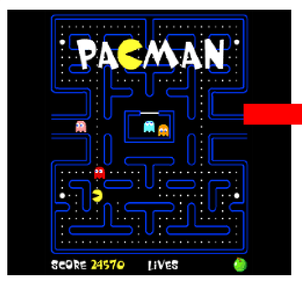
Dans notre exemple, l'analyse statique de l'application malveillante révèle deux problèmes. En premier, les noeuds (`Cancel` et `Ok`) du graphe mènent à la méthode `Connect.open`. En second, l'estimation des valeurs possibles pour l'URL de cette méthode comporte le préfixe de numéro surtaxé (`+338*`).

Pour en savoir plus :
La construction du graphe d'appel et l'estimation des valeurs possibles d'un programme sont des problèmes indécidables en général. Les analyseurs statiques ne savent, donc, construire que des sur-approximations. En conséquence, s'ils trouvent des problèmes, comme dans notre exemple, il faut procéder à une vérification plus fine de cette partie afin de déterminer s'il s'agit d'un risque réel ou si cela est dû à une approximation trop forte. A l'inverse si l'analyseur statique dit qu'il n'existe pas de faiblesse, ceci consiste une preuve de sécurité.

Téléchargement d'applications...

1 → Quels sont les risques ?

2 → Comment y remédier ?



3 → Comment garder l'analyseur à jour ?

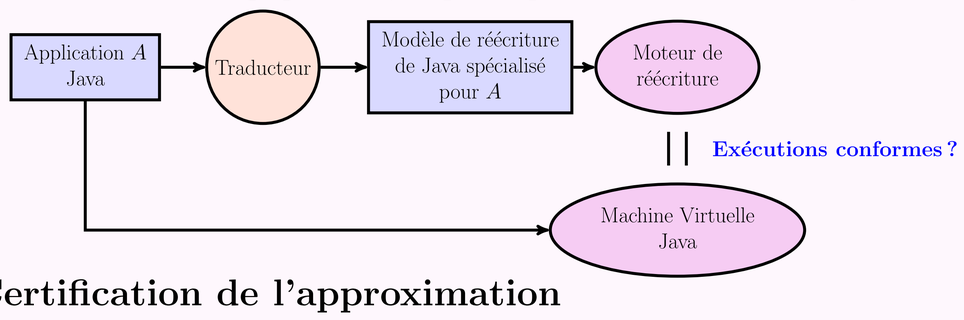
4 → Quelles garanties sur l'analyseur ?

Quelles garanties sur l'analyseur ?

D'une part, il faut assurer la conformité entre le modèle de réécriture et l'application initiale. D'autre part, il faut prouver que l'analyse réalisée sur le modèle de réécriture est sûre. Pour ce faire, on réalise une preuve formelle certifiant que l'approximation est bien un sur-ensemble des états accessibles de l'application à analyser.

1 Conformité modèle/application

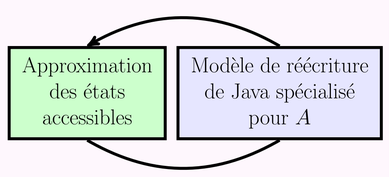
RAVAJ souhaite tirer partie de *l'exécutabilité* des systèmes de réécriture pour tester la conformité des exécutions de réécriture avec les exécutions concrètes. Pour le langage Java, le principe s'instancie de la façon suivante :



2 Certification de l'approximation

Un certificateur d'approximation pour les modèles de réécriture est développé dans l'assistant de preuve Coq. Un tel certificateur sera, comme l'analyseur, *indépendant* du langage et de la propriété cible.

Pour en savoir plus : Le certificateur prouve qu'en appliquant le modèle de réécriture aux états accessibles de l'approximation, on obtient *uniquement* des états déjà représentés dans l'approximation. En d'autres termes, l'approximation est *complète* pour (le modèle de) l'application A.

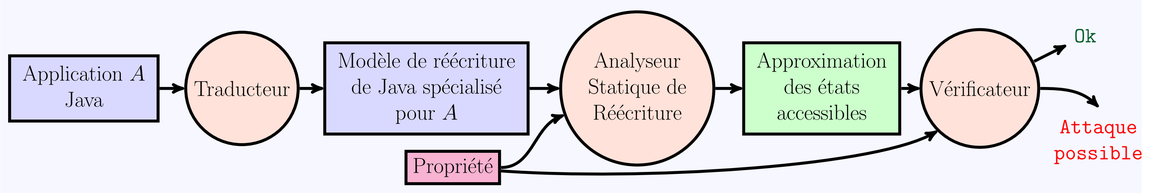


Comment garder l'analyseur à jour ?

Un analyseur est *très dépendant* du langage et des librairies utilisées pour la conception de l'application à vérifier. Le projet ANR **RAVAJ** souhaite dissocier les deux en traduisant l'application et les librairies dans un format intermédiaire, *un système de réécriture*, et en réalisant l'analyse sur ce dernier. L'intérêt est multiple :

- un traducteur est plus simple à réaliser et à mettre à jour qu'un analyseur
- pour des analyses simples, les analyseurs de réécriture sont plus facilement adaptables à la propriété à vérifier que les analyseurs classiques
- il est plus facile de maintenir et d'optimiser *un seul analyseur* dont le format d'entrée n'évolue pas qu'un exemplaire d'analyseur par type ou version d'un langage.

Par exemple, dans le cas du langage Java, le schéma d'analyse est le suivant :



Pour en savoir plus : A partir du modèle de réécriture et de la propriété, l'analyseur de réécriture calcule un sur-ensemble de tous les états accessibles de l'application. Ensuite le vérificateur, s'assure qu'aucun des états du sur-ensemble ne viole la propriété. Comme on ne connaît pas a priori les entrées de l'application, le sur-ensemble des états accessibles peut être non borné. Afin de représenter de façon finie cet ensemble, on utilise des automates d'arbres qui permettent de décrire finiment des ensembles infinis d'états accessibles. Les partenaires IRISA et LIFC de **RAVAJ** ont déjà l'expérience de ce type d'outils formels et de leur utilisation en analyse de programmes (protocoles cryptographiques, byte code Java).

Partenaires **RAVAJ**	Équipe	Laboratoire
Benoît Boyer, Thomas Genet, Thomas Jensen, Vlad Rusu	Lande et Vertecs	IRISA (Rennes)
Pierre-Cyrille Héam, Olga Kouchnarenko	Cassis	LIFC (Besançon)
Emilie Balland, Pierre-Etienne Moreau	Paréo	Loria (Nancy)

Site web : `http ://www.irisa.fr/lande/genet/RAVAJ`
Contact (coordonnateur) : `genet@irisa.fr`
Membres extérieurs : Yohan Boichut LIFO Orléans
Observateur extérieur : Pierre Crégut, France Telecom R&D, AMS/SLE

AGENCE NATIONALE DE LA RECHERCHE ANR