要 旨

情報流仕様に基づいたアクセス制御文の自動生成法

森田 剛正

Java 仮想機械等のプログラム実行系にはスタック検査と呼ばれる汎用的なアクセス制御機構が導入されている．プログラム中, 重要な資源にアクセスする箇所の直前にアクセス制御文を置くことで, 信頼されていないモジュールがその資源にアクセスすることを防止できる．すなわち, 制御がアクセス制御実行文に到達したとき, そのアクセス制御実行文に指定されているアクセス権を実行中のプログラムが保持しているかどうかを検査し, 保持していなければ例外を発生する (または, 強制終了する)．本研究の目的は, スタック検査の拡張である履歴ベースアクセス制御 (HBAC) を対象に, 与えられたセキュリティ仕様を満たすようにアクセス制御文を自動的にプログラムに挿入する方法を提案することである．すなわち, アクセス制御機能の呼び出しのないプログラムとセキュリティ仕様を入力として, セキュリティ仕様を満たすように, 必要なアクセス制御実行文を追加する．本研究では, セキュリティ仕様は情報流に関する仕様として記述し, 情報流解析に基づいて, 仕様に違反する実行系列が発生しないようアクセス制御実行文を生成する．情報流解析とは, プログラムの入力データがどの出力へ流出するかを調べて, そのプログラムが情報漏洩を起こさないかどうかを静的に解析する技術である．本論文では, アクセス権検査文挿入問題の枠組みを定義した後, この問題の co-NP 困難性を示す．また, プッシュダウンシステムのモデル検査法を利用してこの問題を解くアルゴリズムを示し, 試作システムを例題に適用した結果について述べる．

キーワード　言語ベースセキュリティ, スタック検査, 実行履歴に基づくアクセス制御, 情報流解析

Abstract

An Automatic Generation Method of Access Control Statements from Information Flow Specification

Yoshimasa Morita

Stack inspection is a general-purpose access control infrastructure broadly used in such runtime environments as Java virtual machines. Stack inspection prevents untrusted modules from accessing important resources. In this mechanism, a programmer places special permission-check commands just before accessing important resources. When the control reaches a permission-check command, it examines whether the current running process has specified access permissions, and if not, then an exception is thrown or the process is aborted. The purpose of this study is to propose a method for automatically inserting permission-check commands of the History-Based Access Control (HBAC) according to a given security specification, which is an extension of the stack inspection. In this study, the security specification is described as a specification of information flow, and the proposed method generates the access control statements so that any execution sequence dose not violate the information flow specification. This thesis defines the framework of the permission-check command insertion problem and shows the problem is co-NP-hard. After that, an algorithm to solve the problem by using the model checking method for pushdown systems is shown, and the results of experiments where a prototype system is applied to sample programs are described.

key words language-based security, stack inspection, history-based access control, information flow analysis