Verification of Security-Relevant Behavior Model and Security Policy for Model-Carrying Code

Yonglong Wei, Xiaojuan Zheng, Jinglei Ren, Xudong Zheng, Chen Sun, and Zhenhao Li*

School of Software, Northeast Normal University, Changchun, China
5h3ll3x@gmail.com, zhengxj097@nenu.edu.cn, {jinglei.ren, dong128, bbsunchen, zhenhaolee}@gmail.com

Abstract. This article presents a method of verifying the safety of untrusted mobile code, using the model of security-relevant behaviors of code. This method verifies whether models violate users’ security policies to help users decide which programs to run and which not, and hence ensures the security of users’ systems and devices. Based on the framework of model-carrying code, we make several improvements: using the extended pushdown automaton (EPDA) as the program behavior model, reducing ambiguity in regular expressions over events (REE), proposing a new verification algorithm according to above significant improvements.

Keywords: mobile code security, verification, behavior model, security policy, model-carrying code.

1 Introduction

With rapid growth in the use of the Internet and wireless networks, malware, such as viruses, Trojan horses, worms, spyware, are widely spread as hidden in mobile codes and become a dire threat to users’ information security.

In order to meet the challenge concerning mobile code security, R. Sekar et al. proposed the security framework of model-carrying code (MCC) [1]. In spite of the many advantages of MCC method over traditional ones, there are still problems, among which the most significant is the limited precision of the program behavior model and the ambiguity in expressing security policies. To overcome these problems and make MCC method more practical, we make the following improvements to the MCC method: (1) We use the extended pushdown automaton (EPDA) to model security-relevant program behaviors. This new model features in extended attributes including a call stack and state variables which make the PDA more precise so that one sort of impossible paths is eliminated.

* This work was supported by the key project of Science and Technology Planning of Jilin, P.R. China (20080323) and National Collegiate Innovative Experiment Program.
and many mimicry attacks \cite{7} can be detected. (2) We define extended finite state automaton (EFSA) to model the security policy and use standard greedy quantifiers for regular expressions over events (REE) \cite{4}, by which much ambiguity in policy expression is eliminated. (3) We provide a new algorithm for verification of the EPDA-based program behavior model and the EFSA-based security policy.

In order to make MCC framework more applicable to wireless networks and mobile codes. We have also implemented a prototype system on Java Platform Micro Edition (Java ME).

The organization of this paper is as follows: related work are introduced first in Section 2; formal definitions of the program behavior model and the security policy are given respectively in Section 3 and Section 4; in Section 5, procedures of the verification algorithm are given, including analysis of its complexity and proof of equivalence between defined automaton; and Section 6 is the conclusion.

2 Related Work

Researchers have proposed many approaches to ensure the security of execution of untrusted code, mostly based on static analysis or runtime monitor. Prevalent methods include sandbox \cite{5}, code signature \cite{5,6}, proof-carrying code \cite{3}, and java security model \cite{9}. An important common problem with those methods is that they fail to consider the gap between code producers and consumers. Code producers are actually unable to foresee safety requirements of code consumers; the other way round, code consumers cannot determine in advance the proper limits on local recourse access of the program, since the security policy largely depends on program’s function.

R. Sekar et al. proposed the model-carrying code method \cite{1}, providing an ideal safety framework for mobile code execution. The method originates from the domain of intrusion detection. Nevertheless, R. Sekar’s MCC still has defects in several aspects: (1) it uses EFSA as the program behavior model which cannot capture a programs’ function call and return behaviors, and therefore allow impossible paths and possibility of mimicry and evasion attacks \cite{7}; (2) the expression of security policy using REE is ambiguous and imprecise, which may lead to extra false matches when compared with operation sequences and affect the performance of verification.

3 Security-Relevant Behavior Model: EPDA

We employ a new way to identify relations between calls to the same function that are triggered at different points of the program. The automaton pushes a specific symbol to the call stack when the function is called at some point and then pops the symbol when function returns to determine the next state the program reaches after the call. Specifically, we use the EPDA as the model of security-relevant behaviors of code, which is formally defined as follows: