Component-Based Access Control: Secure Software Composition through Static Analysis*

Pierre Parrend and Stéphane Frénot

INRIA ARES / CITI, INSA-Lyon, F-69621, France
Tel.: +334 72 43 71 29; Fax.: +33 472 43 62 27
{pierre.parrend, stephane.frenot}@insa-lyon.fr

Abstract. Extensible Component Platforms support the discovery, installation, starting, uninstallation of components at runtime. Since they are often targeted at mobile resource-constrained devices, they have both strong performance and security requirements. The current security model for Java systems – Permissions – is based on call stack analysis. This is very time-consuming, which makes it difficult to use in production environments.

We therefore define the Component-Based Access Control (CBAC) Security Model, which emulates Java Permissions through static analysis at the installation phase of the components. CBAC is based on a fully declarative approach that makes it possible to tag arbitrary methods as sensitive. A formal model is defined to guarantee that a given component have sufficient access rights, and that dependencies between components are taken into account.

A first implementation of the model is provided for the OSGi Platform, using the ASM library for code analysis. Performance tests show that the cost of CBAC at install time is negligible, since it is executed together with digital signature verification which is much more costly. Moreover, unlike Java Permissions, the CBAC security model does not have any runtime overhead.

1 Introduction

Extensible Component Platforms enable the composition of components which are provided by several issuers and that can be loaded, installed and uninstalled at runtime. In the Java world, such platforms can be Java Cards [17], MIDP [12], or the OSGi platform [14]. The functional composition is supported for instance, in the case of the OSGi Platform, through efficient support of dependency resolution. Composition of non-functional properties, such as security, are so far not supported in such systems.

To support the composition of access rights of components, we propose the Component-based Access Control (CBAC) Security Model. The objective is to

* This work is partially funded by MUSE II IST FP6 Project n026442.
replace Java Permissions through validation of the access rights via static code analysis. Permissions prove to be difficult to use in production environments due to the excessive overhead they imply at runtime\textsuperscript{1}. To reduce this, we take advantage of the installation phase of the bundles to perform suitable checks to free the runtime phase from costly checks. The advantages of this approach based on static-analysis are numerous. The main ones are the gain of performance at runtime, the absence of program interruption, and the flexibility of policy expression. The latter is fully declarative, which makes it possible to protect additional JVM methods (Threads, ClassLoader) and component methods.

This paper is organized as follows. Section 2 presents the existing security models for Java Component Platforms. Section 3 presents the CBAC Security Model. Section 4 presents the validation of our approach. Section 5 concludes this work.

2 Security Models for Java Component Platforms

Since it has been designed with security in mind and since it has been subjected to extensive testing by the community after its initial releases, the Java Virtual Machine \textsuperscript{13} is usually considered a very secure execution platform.

Various security mechanisms have been proposed to build secure Java Platforms. In this section, we review them to identify whether they provide suitable solutions for Component Platforms. First, the J2SE security mechanisms are presented. Then, optimizations for access control through static analysis, which limit verification overhead at runtime, are discussed. Lastly, security mechanisms for Component Systems are presented.

2.1 Java Security

The security properties of the Java Component Platform are supported by the Virtual Machine itself, which provides a platform that is able to fully isolates the applications from their environment. These mechanisms build a sound basis to support a specific Access Control mechanism based on method calls stack inspection: Stack-based Access Control (SBAC).

The Java Virtual Machine is characterized by following built-in security features that makes it safe: type safety, automatic memory management, Bytecode validation and secure class loading.

Type Safety ensures that programs cannot contain type mismatches, and cannot execute elements that are not functions \textsuperscript{21}. It prevents the use of pointers to execute arbitrary code and to prevent buffer overflows.

Automatic memory management is performed through Garbage Collection. It prevents memory leaks and thereby relieves the developer from memory management, which is error prone.

Bytecode validation consists of checking the compliance of the executable code to the JVM specification before it is loaded for execution. In particular,

\textsuperscript{1} Our tests show that the performance loss amounts between 55\% and 144\% for specific calls.