Poster Extended Abstract, EuroSys Conference, Leuven, Belgium, April 18-21 2006.

# Title: Dependability for Component Systems Deployment

Name: Pierre Parrend
E-mail: pierre.parrend@insa-lyon.fr
Student: yes (Ph.D.)
Affiliation : INRIA Ares Team (France)

Themes : Middleware, Security, Management

Operating Systems and Server Platforms tend to be increasingly organized as sets of components, which enables greater flexibility, as well as easier system upgrade. Even though such architectures evolve more easily as monolithic ones, they are also more difficult to secure. Bringing new components in these kind of systems means also not being able to always assert component reliability and innocuousness  We conduct experiments on component system dependability validation using the OSGi platform in the frame of the IST Muse Project for wide band Service at Home delivery. OSGi provides an execution and life cycle management environment for embedded applications in Java [OSGi05].

Deployment is the first step of component management life cycle, after development is completed . It is made up of component publishing, discovery, dependency resolution, downloading, installation, configuration, starting [Hall99]. Dependability [Avizienis00] of a system is the conjunction of several properties of systems : availability, reliability (continuity of correct service), safety (absence of catastrophic consequences), confidentiality, integrity, maintainability. Dependability for component deployment can thus be defined as 'Load and install the right component at the right time'. The right component is one that the user knows not to be malicious, and which provides the desired service. As far as each user do not know a priori each component, two criteria can be applied in order to check that a component is not malicious : either the component is trusted, or it is possible to assert that the code can not execute dangerous actions (or combination of both).

The first step to provide a secure component system is to ensure that deployed components come from a trusted provider. This is done through cryptographic signature. Two different technologies exists : signature validation through Public Key Certificate (X.509) Path, often used in closed systems (where a Certification Authority provides trustworthy certificates), or through PGP and publicly available key servers, often used for open Operating Systems based on components (also named modules, or packages). RPM, and Gentoo Portage notably use this approach.
Second step is to ensure safe execution of installed components. Extreme case is Java Sandboxing, where all actions that could possibly be harmful are blocked. This enables untrusted applets to be executed safely.  In most cases however, it is necessary to provide access to the file system, or to the network, but not to the whole system. This is achieved through execution permissions. The Spin OS, based on the Modula-3 language, or VINO OS, use this mechanism. OSGi also provides permissions for controlling component interoperability that are used together with Java permissions.

Our work targets the design and development of a dependable OSGi framework. Few work having been done – surprisingly enough – about this subject, it is necessary to provide a mean of validating OSGi components (named bundles) at load time. OSGi specifications propose to sign bundles through X.509 Signature Certificates, which are included in the archive. This permits validation of the integrity of the bundle, and to authenticate the signer(s). However, confidentiality is not possible, because bundle signing specifications consider that one must be able to use a bundle without taking care of the signature. This is not a problem in open source projects – where the code is anyhow available – but builds a severe restriction in closed systems, enterprise systems, or Service at Home services delivery. We propose two solutions : the first one is to put the encrypted component in the signature file. This implies only minor changes to OSGi specifications and prevents unauthorized users from installing the bundle. The other requires the existence of one (or several) centralized repository(ies). It is then possible to connect the client and the server through a secure communication protocol (such as HTTPS, SSH, or over Virtual Private Networks). In this configuration, OSGi security specifications are  too heavy.

The second aspect of bundle validation at load time is to ensure that permissions that are necessary for correct execution do not go against existing policies, and that the code does not contain security leaks. Guaranteeing that permissions are valid in the Java world is normally done at runtime, and can cause misbehavior of the program. We propose a new permission validation protocol that enables permission checking before bundle download, thus preventing performance downfall due to loading of unsuitable code: first, a bundle description file containing policy parameters required for its correct execution is downloaded by the client. If this policy is compatible with actual client security policy, the bundle itself is downloaded. Then, coherence of code and description file is checked. It can be extended with Proof Carrying Code (PCC) for guaranteeing code harmlessness.
Realizing experiments on dependable component systems on the OSGi platform proves to be highly valuable as well for general analysis of middleware platforms as for the study of OSGi itself, which specifications seem to need to adapt to real network systems – and not only to closed world. This ongoing work is still to be completed by performance analysis of the different solutions. Being specified as a stand-alone platform with some component life cycle management facilities (validation, installation, start, ...) but without any consideration on how these components are downloaded, OSGi framework seems to have still a good evolution potential.

**[OSGi05]** OSGi Service Platform, Core Specification Release 4, OSGi Alliance, 2005.
**[Hall99]** R.S. Hall, D. Heimbigner,  & A.L. Wolf, A Cooperative Approach to Support Software Deployment Using the Software Dock, ISCE 1999.
**[Avizienis00]** A. Avizienis, J.C. Laprie & B. Randell, Fundamental concepts of dependability, Technical Report LAAS (Toulouse, France), 2000.