Component Middlewares that are extensible at runtime have very promising applications in two different worlds: first, embedded systems rely on them to support adaptability to their context or to specific uses; secondly, application servers take advantage of the easy component management they provide, in particular by enabling system upgrade without full system reboot. Extensible Component Middleware enable these systems to load code from their environment, to install them and to start them seamlessly. Dramatic functionnal and management-level improvement are thus made possible.

However, such a possibility opens a brand new attack vector: since the loaded code is not necessarily provided by the system manager, it may either be malicious or simply unstable, and compromise the expected behavior of the component platform.

Providing a safe environment for runtime extensibility implies solving two complementary problems. First, code that is loaded over insecure networks needs to be protected from malicious agents that could eavesdrop or even modify it. Secondly, the component platform and the applications that are running on it must be protected against the newly loaded code.

My works takes place in the context of the INRIA-ARES Team development which aims at providing a management environment for the OSGi Platform. I therefore concentrate on dependability concerns relative to the OSGi Platform, and validate my contributions by integrating them with the existing management tools.

**Context**

This study is conducted in the context of the MUSE IST Project n°026442. The objective of the project is to improve the state of the art of Home Gateways, that provide at the same time network access and personalized services [festor06residentialGateway], [parrend06securAnalysis]. Other applications of OSGi Gateways are automotive services or personalized mobile services [parrend07privacy], Application Servers such as IBM Websphere 6.1, or the Jboss application server, are not considered more specificaly, but may nevertheless benefit from my research.

**Secure Deployment**

Securing the deployment of OSGi bundles means that the life-cycle of bundles must be well understood, and that all phases between the publication and the installation on the execution platform must be protected from malicious third party. I first implemented tools to support secure deployment of the OSGi bundles – which was not yet available, and then based two research studies on this implementation.

The life-cycle of a bundle is made up of the following phases: the development, the packaging, the publication on a Bundle Repository, are the phases that are supervised by the Issuer. The bundle discovery, selection, download, installation, and start-up are the phases that are supervised by the client platform. According to the OSGi specification, and current practices, the security is achieved by digitally signing the bundles between the packaging phase and the publication phase. The issuer is responsible for this signature. The client Platform must check the validity of this signature before installing the bundle, i.e. it verifies 1) that the digital signature has
not been tampered with, and 2) that the Issuer is a known and trusted one.

To support this process, I developed two tools. The first tool is an OSGi Release 4 compliant implementation of the digital signature and validation process of bundles. It is meant in particular to replace the Felix digital signature validation process, which is not R4 compliant. The code is available as an Open Source project, Sfelix [Sfelix], under the version 0.1. It is an extension of the Felix platform that integrates my digital signature validation implementation. The second tool support the publication phase of the bundle life cycle: SF-Jarsigner [SF-Jarsigner] encapsulates the digital signature generation process along with a convenient tool for publishing the signed bundles on a Bundle Repository. Currently, only the FTP protocol is supported for publication. Once published, the bundles can be discovered and downloaded with an OBR2 compliant tool, as the BundleRepository tool which is provided in the Apache Felix project. If the clients are built with my Sfelix implementation, the SF-Jarsigner tool provide a specification compliant security for OSGi bundles. Since OSGi digital signature validity criteria are stronger than default Java Archive ones [parrend07secureDeployment], my tool can also be used together with other OSGi platforms. Note that OBR1 repositories are currently not supported, which cause the Knopferfish client not to be compatible with it. Be careful when using OSGi platforms with security on, all implementations do not comply with the specifications!

This development work provide us with a unique support for Life-Cycle long secure deployment process. It enable to study the required management mechanisms for bundle deployment.

The first study that is based on this implementation is the integration of Sfelix and SF-Jarsigner together with the OSGi management tools of the INRIA-ARES Team, namely the MOSGi and VOSGi projects. MOSGi provides a console based on JMX agents to remotely manage OSGi platforms. VOSGi enables Virtual OSGi instances to be run inside a main OSGi platform, thus supporting multi-provider services. This functionalities have been developed to support the management of Home Gateways. So as to handle the notification of bundle rejection because of unvalid digital signature, the Bundle Life-Cycle inside the OSGi Gateway has been extended with a new state: when an invalid bundle is installed, it goes in the ‘REJECTED’ state before being cleanly removed. This enables the tracking of unsuccessfull installation, which can reveal malicious actions.

The second study that is based on my implementation of the secure deployment process is the modification of the digital signature process [parrend07ibcrypto]. I propose to use Identity-based Cryptographic mechanisms, instead of the default RSA/SHA-1 digital signature. This modification has radical management implications. Since Identity-based Cryptography enables clients to deduce the public key from the name of the Issuer, and from some cryptographic parameters of the Public Key Generator (PKG), they can quickly check whether the private key used for signature has been issued by the trusted PKG. Consequently, the clients do not need to connect to the Certification Authority to check a digital signature. The PKG must only handle the connections of some dozens of Issuers for very large systems, whereas a PKI-infrastructure would need to support the connection of a couple of millions of Home Gateways in the context of ADSL providers. Financial benefits are thus important. Moreover, since a time-stamp is usually appended to the Identifier, keys can be updated frequently (e.g. Daily), and no Certificate Revocation mechanism is required. It is sufficient for the PKG not to issue new keys for the revoked issuer. Side benefits also come from the reduced size of keys and certificates because of the different algorithms used.

Further work is needed about secure deployment to propose a complete and innovative security management infrastructure for OSGi Bundles.

Assessment for Secure Component Execution

Controlling the deployment process is necessary to protect our OSGi infrastructure from external hackers. However, it does not provide any guarantee related to the execution safety of the installed Bundles. All it provides is that one knows the Bundle Issuers. In the context of extensible platforms, which build their adaptability on the possibility of installing code from third party Issuers that are potentially not known beforehand, this is clearly insufficient. Weaknesses of the OSGi
platforms need to be identified, and suitable solutions proposed. This is done is three steps: first, a method is defined, so as to formalize the identification of the weaknesses and the exploitation of this knowledge; secondly, weaknesses need to be identified; lastly, solutions are to be proposed and validated. The conclusion of this study is a set of recommendation of evolutions for the OSGi Platform Specification [parrend07osgiVulnerabilities].

The method for identification and analysis of the weaknesses of extensible component platforms is the following. It is based on 'Semi-formal Vulnerability Patterns', i.e. Patterns that document in a systematic way the properties of the vulnerabilities using system-specific taxonomies. First, a system model is defined, to identify the characteristics of the system under study. Next, a Semi-Formal Vulnerability Pattern is defined, that gathers four types of informations: Vulnerability Reference, with Name, Identifier, and taxonomy-based characterization; Description, to express more precisely the specificities of the vulnerability; Implementation, with the concerned platform implementations, and test coverage; and Protection, to identify the existing and potential security mechanisms that can be used to patch this vulnerability. Next, a catalog is made with the vulnerability I identified. Lastly, analysis if performed thanks to the taxonomies I defined, and recommendations for improvement are made.

The system model for the OSGi Platform is made up of two complementary part. First, the execution platform itself is modeled and, secondly, the bundle format. This modelisation is a quite simple one, and is directly deduced from the specification. However, it must be coherent with the system-specific taxonomies, and must therefore be explicit. The OSGi execution Platform is defined as follows. It is made of three main layers, the resolution, life-cycle and service layers. An additional module is to be taken into account, namely the Bundle Downloader module, which supports bundle discovery and download. Two types of interactions are to be considered: between the bundles and (one of the four layers of) the platform, and between the installed bundles. This various types of interaction can be vulnerability-prone. The OSGi Bundle format is the following. It is a Java Archive, with an extended Manifest, digital signature similar to the one of the Java archives (only verification criteria differ), and ressources, with are mainly java classes. Code that exploit the vulnerabilities of the platforms can be found in any of these elements.

The Semi-formal Vulnerability Pattern is made of a series of fields that are usefull to characterize with precision the vulnerability. Our Pattern differs from existing one in that it aims at precise analysis and expression of the (potential) solution, whereas current ones are minimal, and aim at user information rather than on the expression of technical data. Several fields are defined through a system-specific taxonomy, namely the Location of the Exploit Code (here, in the bundle), the Source of the Vulnerability (here, in the platform), the Target of the Attack, the Type of Consequence of the attack, the Introduction Time of the Exploit Code, the Exploit Time. Existing and potential protection mechanisms are also identified, as well as reactions to the attack.

Each vulnerability must be characterized according this Semi-formal Vulnerability Pattern. I implemented each vulnerability in the Malicious-Suite Development project [Malicious-Suite], so as to validate the theoretical vulnerabilities, and to enable tests on various implementations of the OSGi Platforms. 30 Vulnerabilities have been identified. Several Open Source OSGi implementations have been tested: Concierge, Equinox, Felix, Knopflerfish.

One of the main benefit of our analysis is to provide a detailed view of the requirements for new security mechanisms, or better support for existing ones. Three categories of security mechanisms are identified. The first one is the modification of the OSGi Platform so as to enhance its robustness. The second one are dangerous functions that should be protected through Java and OSGi Level Permissions. Little convenient tools currently exist to manage them. Lastly, the most promising and most complex security mechanism is the development of code analysis tools that could detect potentially malicious code at installation. One other conclusion of our study is that half of the vulnerability come directly from the OSGi platform, and the other half is related to the underlying Java Virtual Machine. Consequently, our study can be exploited in context other than the OSGi Platform.

Based on the identified OSGi Vulnerabilities, I developed Sflex, which is a hardened OSGi implementation build on the Apache Felix Project. These improvements are expressed as recommandations for improvement of the OSGi Specifications. Some JVM vulnerabilities are also identified in the Sun JVM 1.6 and in the Gnu_Classpath Open Source implementation of the Java classes.

The use of Semi-formal Vulnerability Patterns provides a framework for vulnerability
characterization and analysis. It also provides a qualitative comparison of existing platform implementations. However, it falls short in evaluating the relative severity of the attacks, and to provide evidence so as which specific attacks should be protected from in priority. I therefore initiate a study so as to identify the metrics that can be used to provide a quantitative assessment of the vulnerability status of execution platforms [parrend07metrics]. The goal is to identify 1) the attacks that have an important impact on the system and 2) the protection mechanisms that can be used with benefit. The main idea is to take the consequence of attacks on the system into account. This approach should allow to balance the actual benefit of protection mechanisms against their cost, being the cost in term of functionalities (Java Permissions provide a great improvement in security by drastically limiting the available functions) or in term of development cost (to evaluate security mechanisms that are not yet available).

**Pervasive Services and data privacy**

Secure execution environments are useful if they can be used in conjunction with innovative services. I therefore performed several prospective studies to explore the relationship that can be built between OSGi-based environments and new applications of mobile and pervasive computing.

In the first study, I propose to open the Home Gateway to bring the control back to their user. I believe that such an evolution is a natural evolution of Home Gateways, that were at first closed worlds, and that now begin to be open to third party service providers. Current systems do not provide sufficient resource isolation and execution safety to support our proposition, but new Business Models could soon emerge, for instance by creating communities that could share resources through their Gateways, without needing to keep their Home Computer on [parrend06communities].

The second study is focused on the handling of privacy data in pervasive systems. OSGi is a promising execution environment for such applications, since it is designed for resource constrained devices. However, using mobile services also implies providing significant data to take advantage of these services. Together with the Institut für Telematik der Universität Freiburg, we propose a first exploration of the required architecture and privacy data model to enforce privacy in pervasive systems [parrend07privacy].
Logiciels

Projets de développement avec objectif de recherche: (Disponibles sur INRIA SourceForge)
[SFelix] SFelix: version sécurisée de la plateforme Felix (http://sfelix.gforge.inria.fr/)
[SF-JarSigner] SF-JarSigner: Signature et Déploiement de bundles OSGi (http://sf-jarsigner.gforge.inria.fr/)
[Malicious-Suite] Malicious Suite: Composants malicious exploitant des vulnérabilités d'OSGi

Développements annexes:
[Component-Gui] ComponentGui: un support simple de plugins basé sur OSGi
[OSGi-Archetypes] OSGiArchetypes: archetypes Maven pour la génération de squelettes de bundles OSGi
[Disponibles sur INRIA SourceForge].

[SF-Monitor] SF-Monitor, un outil de monitoring de Plateformes OSGi (Disponibles sur INRIA SourceForge)
[Doc-Suite] DocSuite, ensemble d'outil de génération de documentation à partir de fiches techniques au format
XML (Xm22Tex: générateur de fiches Latex; XmAnalyster: Xm22TexTable: générateur de tableau récapitulatif;
XmAnalyster: générateur de graphiques statistiques)

Publications

2007

[parrond07ibcarypto] Identity-Based Cryptosystems in the OSGi Service Platform, Samuel Galice, Pierre
Parrend, Stephane Frenot, Stephane Ubeda, International Conference on Emerging Security Informations,
Systems and technologies, IARIA SecuWare 2007, Valencia, Spain, October 2007.
[parrond07privacy] Privacy-Aware Service Integration, Pierre Parrend, Stephane Frenot, Sebastian
Hoehn, Services Integration in Pervasive Environments, Istanbul, Turkey, July 2007.
[parrond07secureDeployment] Supporting the Secure Deployment of OSGi Bundles, Pierre Parrend,
Stephane Frenot, First IEEE WoWMoM Workshop on Adaptive and Dependable Mission- and byss-critical
mobile Systems, Helsinki, Finland, 18 June 2007.

2006

[parrond06secaAnalyzed] A Security Analysis for Home Gateway Architectures, Pierre Parrend and
Stephane Frenot, International Conference on Cryptography, Coding & Information Security, CCIS 2006,
November 24-26, Venice, Italy.
[parrond06osgideployment] Parrend, P. & Frenot, S., Secure Component Deployment in the OSGi(tm)
[parrond06communities] Service-Oriented Distributed Communities in Residential Environments,
Pierre Parrend, Yvan Royon and Noha Ibrahim, 1st IEEE International Workshop on Services Integration
in Pervasive Environments June 29, 2006, Lyon, France.
[parrond06dependability] Dependability for Component Systems Deployment, Pierre Parrend,

2005

[parrond05ontologies] Use of Ontologies as a Way to Automate MDE Processes, Pierre Parrend,
[parrond05mde] MDE et CSCW Groupware Travail Coopératif capillaire, Pierre Parrend, Master's

2004

[podary04realTim] Comparison and temporal validation of automotive real-time architectures, K.
[parrond04validation] Validation temporelle d'architectures embarquées pour l'automobile, P. Parrend
and I. Augé-Blum. Technical Report, CITI Lab, INSA de Lyon, July 2004

Soumise (travaux en cours)

[parrond07osgiVulnerabilities] Vulnerabilities of the OSGi Platform - An Experimental Classification,
[parrond07metrics] A Set of Metrics for Security Assessment in Component-based Software Systems,
Pierre Parrend, Stephane Frenot, International Conference on Dependable Systems and Networks, Florence,
Italy, June 2007.

Autres délivrables réalisés

[fe sto06residentialGateway] Deliverable D B3.4 - Specification of Residential Gateway configuration,
Edited by Olivier Festor, Sam D'Haesseler, 23/03/2006, IST Project n°026442 Muse.