A Dependability Framework for building secure Dynamic Component Systems

Pierre Parrend
Pierre.parrend@insa-lyon.fr
Lab. CITI, 21, Avenue J. Capelle
69621 Vileurbanne Cedex
Introduction

• 2 Types of Component Systems
  – Static Components
    • Guaranteed Properties, Strong Specifications, QoS
    • Ex: Paladio
    • Design Time Validation
  – Dynamic Components
    • Runtime extension (and removal)
    • Handheld, Embedded and Automotive Systems, Application Servers
    • Ex: JavaCard, MIDP, OSGi in the Java world
    • Install Time Validation
Introduction

• Security Models for Extensible Component Middleware
  – Identify the Issues
  – Propose first Solutions
  – Make (some) Tools available

• Target System
  – OSGi Platform over Java
    • Even broader use
    • Likely to be integrated in Java 7
    • Object study of the Middleware Team of the CITI Lab.
Summary

- **Extensible Component Platforms**
- Dependability for Extensible Component Systems
- Secure Deployment
- Secure Execution
- Achievements and Open Issues
Extensible Component Platforms

- Structure

Remoting

- Remotely available Services

Code Level (Inside the platform)

- Local interactions

System Level (local to the host; outside the platform)

- Local execution

Component Repository

Deploy
Extensible Component Platforms

• Example 1: MIDP
  • Mobile Information Device Profile
  • Defined by Sun
  • Applications
    • Middlet Suites
    • Defined in an external JAD File
      • Java Application Descriptor

![Diagram showing MIDP and CLDC layers with Middlet Suites 1, 2, and 3]
Extensible Component Platforms

- Example 2: OSGi
  - Was 'Open Service Gateway Initiative'
    - Is now an adjective
  - Forstered by the OSGi Alliance
  - The Platform - The Bundles
Summary

- Extensible Component Platforms
- **Dependability for Extensible Component Systems**
- Secure Deployment
- Secure Execution
- Achievements and Open Issues
Dependability for Extensible Component Systems

- The Dependability Approach to Security

- Motivation
  - A robust system must withstand Attacks
  - A secure system must withstand Faults

  ![Diagram](image)

- Both aspects must be considered when building a real system
- Both aspects must are the two faces of the same coin
  - different techniques
  - different communities
  - same target systems
Dependability for Extensible Component Systems

- Protection against Attacks
  - For Dynamic Component Systems

<table>
<thead>
<tr>
<th></th>
<th>Deployment</th>
<th>Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attack Prevention</strong></td>
<td>Digital signature</td>
<td>Intrusion Benchmarking</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Code validation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Access Control</td>
</tr>
<tr>
<td><strong>Attack Tolerance</strong></td>
<td>Network IDS</td>
<td>Host IDS</td>
</tr>
<tr>
<td><strong>Attack Removal</strong></td>
<td>Protocol Patching</td>
<td>Platform patching</td>
</tr>
<tr>
<td><strong>Attack Forecasting</strong></td>
<td>Audit based on network-level</td>
<td>Audit based on host-level</td>
</tr>
<tr>
<td></td>
<td>Security logs</td>
<td>Security logs</td>
</tr>
</tbody>
</table>
Dependability for Extensible Component Systems

- Dependability Properties for Execution
  - For Dynamic Component Systems
  - First Consider Prevention

<table>
<thead>
<tr>
<th></th>
<th>Fault Prevention</th>
<th>Attack Prevention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>Prevent stopping/destruction of Platforms and Components</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>Prevent Erroneous Method Calls</td>
<td></td>
</tr>
<tr>
<td>Safety</td>
<td>-</td>
<td>(No full guarantee on Java-based Systems)</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Ressource Isolation</td>
<td></td>
</tr>
<tr>
<td>Integrity</td>
<td>Access Control</td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>Component Life-Cycle by default in OSGi</td>
<td>Separation of Management and Applicative level required</td>
</tr>
</tbody>
</table>
Dependability for Extensible Component Systems

- The Opportunities with Java-based Systems
  - JVM is designed to be safe

- The Problem with Java-based Systems
  - Components have not been taken into account when designing Java Security
  - Complex Systems
  - Constraint unaware developers
    - The community likes to build functional stuff
    - The community does not like security, pre/post conditions

An easy-to-use solution is required for Java Extensible Component Platforms
Dependability for Extensible Component Systems

• The Trust Model for Extensible Component Platforms
  – Trusted Platform, Untrusted Network
  – Components with trusted origin, issuer does his best to provide 'satisfactory' components
  – 'Satisfactory' is highly subjective – characterization is needed (performance, safety, security, etc.)
Summary

- Extensible Component Platforms
- Dependability for Extensible Component Systems
- **Secure Deployment**
- Secure Execution
- Achievements and Open Issues
Secure Deployment

The Deployment Process in the OSGi Platform

1) Publication
2) Discovery
3) Download
4) Installation
5) Start

Deployment Stakes
- Threats on Deployment
- Key Management
Secure Deployment

- Threats on Deployment
Secure Deployment

- Key Management

Diagram:
- Sign Bundles with JarSigner
- Publish to Service Issuer
- Register with PKI Authority (ACS)
- Check Identity
- Validate Bundle with Security Layer
- Install on SFelix Client Platform
Secure Deployment

• Required Development
  – OSGi Platform that is able to check Digital Signature
  – Bundle Signer and Publisher
  – Not a research topic, but an urgent requirement!
Secure Deployment

- **SFelix – Secure OSGi Platform**
  - [http://sfelix.gforge.inria.fr/](http://sfelix.gforge.inria.fr/)
  - SFelix v0.1
    - OSGi Release 4 Implementation of the Bundle Signature Validation Process
    - Beware of JVM-only solutions!
  - SFelix v0.2
    - Robust against ill-coded Bundles
    - Code is not yet released publicly
Secure Deployment

- SFelix – Secure OSGi Platform

```java
-> obr start "HTTP Service"
target resource(s):
------------------
HTTP Service (0.8.0.SNAPSHOT)

Deploying... Resolver: Install error - org.apache.felix.http.jetty
org.osgi.framework.BundleException: Could not create bundle object.
  at org.apache.felix.framework.Felix.installBundle(Felix.java:1347)
  at org.apache.felix.framework.Felix.installBundle(Felix.java:1322)
  at org.apache.felix.framework.BundleContextImpl.installBundle(BundleContextImpl.java:90)
  at org.apache.felix.bundlerepository.BundleRepositoryImpl.deploy(BundleRepositoryImpl.java:57)
  at org.apache.felix.bundlerepository.BundleRepositoryImpl.deploy(BundleRepositoryImpl.java:396)
  at org.apache.felix.bundlerepository.CommandImpl.deploy(BundleRepositoryImpl.java:294)
  at org.apache.felix.bundlerepository.CommandImpl.execute(BundleRepositoryImpl.java:168)
  at org.apache.felix.shell.impl.Activator$ShellServiceImpl.executeCommand(Activator.java:233)
  at org.apache.felix.shell.impl.ShellUI.run(ShellUI.java:169)
  at java.lang.Thread.run(Thread.java:656)
Caused by: org.osgi.framework.BundleException: Bundle: unsecure
  at fr.inria.ares.framework.cache.DefaultSecureBundleArchive.checkArchiveValidity(DefaultSecureBundleArchive.java:73)
  at org.apache.felix.framework.Felix.installBundle(Felix.java:1323)
  ... 8 more
```

29/10/2007
Secure Deployment

- SF-JarSigner – Bundle Signer and Publisher
  - http://sf-jarsigner.gforge.inria.fr/
  - The Archive Analysis Panel
Secure Deployment

- Secure OSGi Deployment and Java Security

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Sun Jarsigner</th>
<th>Java with Security Manager</th>
<th>Felix</th>
<th>S-Felix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsigned Archive</td>
<td>W</td>
<td>A</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>Unknown Signer</td>
<td>A</td>
<td>A</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>Addition of Resource</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>R</td>
</tr>
<tr>
<td>Removal of Resource</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>R</td>
</tr>
<tr>
<td>Modification of Resource</td>
<td>R</td>
<td>R</td>
<td>W</td>
<td>R</td>
</tr>
<tr>
<td>Invalid Order of Resources</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>R</td>
</tr>
<tr>
<td>Signature of Embedded Archive Invalid</td>
<td>R</td>
<td>R</td>
<td>W</td>
<td>R</td>
</tr>
<tr>
<td>Time of Check</td>
<td>Test</td>
<td>Exec</td>
<td>Exec</td>
<td>Install</td>
</tr>
</tbody>
</table>

A: Accept; R: Reject; W: Warning;
Summary

- Extensible Component Platforms
- Dependability for Extensible Component Systems
- Secure Deployment
- **Secure Execution**
- Achievements and Open Issues
Secure Execution

• Why is OSGi a potentially very secure Platform
  – Java is designed to be secure
    • Type Safety
    • Garbage Collection
    • Bytecode verification
    • Secure class-loaders
  – OSGi provides a sound programming model
    • Namespace isolation between bundles
    • Very convenient management model
      – 'Maintainability' properties of the Dependability Framework
    • Anchor for install time checks
Secure Execution

• Why is OSGi currently NOT a secure Platform
  – No serious development effort from the community
  – Designed for secure single app execution
  – Opens a new attack Vector
    • Dynamic installation of unknown (malicious ?) code
    • Not an issue in Open Source projects
      – Everybody is benevolent
      – Ready to bet NO Eclipse plugin contains malware ??
    • Not an issue in Industrial projects
      – Never load untrusted code
      – Never load confidential code over the Internet ??
    • Not an issue until one really needs Security
  – First Requirement
    • Identifies the weaknesses
Secure Execution

- **OSGi Vulnerabilities - Identification**
  - Black Hat Approach
  - How to get
    - Denial of Service
    - Data/Code exposure
    - Erroneous output
  - Take the Specification and Code Malicious Bundles
    - That exploit the Java framework
    - That exploit OSGi specific features
Secure Execution

- OSGi Vulnerabilities – Identification
  - Recursive Thread Creation

```java
public class Stopper extends Thread {
    Stopper(int id, byte[] payload) {
        this.id = id;
        this.payload = payload;
    }

    public void run() {
        System.out.println("Stopper id: "+id);
        Stopper tt = new Stopper(++id, payload);
        tt.start();

        Stopper tt2 = new Stopper(++id, payload);
        tt2.start();

        Stopper tt3 = new Stopper(++id, payload);
        tt3.start();
    }
}
```
Secure Execution

- OSGi Vulnerabilities – Classifications
  - Security Taxonomies for the OSGi Component Platform
Secure Execution

- OSGi Vulnerabilities – Classifications
  - Security Taxonomies for the OSGi Component Platform

Diagram:
- Vulnerability Source
  - Operating System
    - Runtime
      - Kill the Platform
  - JVM
    - APIs
  - OSGi Platform
    - Module Layer
    - Life-Cycle Layer
    - Service Layer
    - Bundle Repository
      - Client
    - Application Code
      - Bundle Oversize
      - Infinite Loop
      - Excessive Service Registration
      - Zombie Data
      - Duplciate Package Import
      - Data Modification - Reflection
      - Halt the Platform
      - Oversize
Secure Execution

- OSGi Vulnerabilities – Classifications
  - Security Taxonomies for the OSGi Component Platform

Diagram:
- Location of Exploit Code
  - Bundle Archive
  - Manifest
  - Activator
  - Application Code
    - Native Code
    - Java Code
    - Java API
    - OSGi API
      - Bundle Fragment
  - Bundle Oversize
    - duplicate import package
    - Hanging Activator
    - Kill the Platform
    - Infiinite Loop
    - Recursive Thread Creation
    - Excessive Service Registration
    - Fragment Substitution

29/10/2007
Dependability of Component Platform
Secure Execution

- OSGi Vulnerabilities – Classifications
  - Security Taxonomies for the OSGi Component Platform

```
Attack Target
/
/   
/    
Platform ---------- OSGi Element
                  /
Platform Management Utility  Bundle  Service  Package

Unavailability  Hanging Activator  Pirat Manager  Cycle between Services  Component static data Modifier
```
Secure Execution

- OSGi Vulnerabilities – Classifications
  - Security Taxonomies for the OSGi Component Platform

```
Consequence Type

Java, OS, Framework

Unavailability
Performance Breakdown
Undue Access

system.exit(0)
Infinite Loop
Code Observer
```
Secure Execution

- OSGi Vulnerabilities – Characterization
  - The Semi-formal Vulnerability Pattern for the OSGi Extensible Component Platform
  - Reference
    - Vulnerability Pattern (VP) Id
    - Taxonomy-based characterization
  - Description
    - More Text
  - Protection
    - Actual Protection
    - Potential ones
  - Implementation
    - Robust and Vulnerable platforms
    - Implementation case coverage
Secure Execution

- OSGi Vulnerabilities – Characterization
  - Freezing Management Utility – Hanging Thread

Vulnerability Reference

- **Vulnerability Name:** Management Utility Freezing - Thread Hanging
- **Extends:** Hanging Thread
- **Identifier:** Mb.osgi.5
- **Origin:** Ares research project ‘malicious-bundle’
- **Location of Exploit Code:** Bundle Activator
- **Source:** OSGi Platform - Life-Cycle Layer (No safe Bundle Start)
- **Target:** OSGi Element - Platform Management Utility
- **Consequence Type:** Unavailability
- **Introduction Time:** Development
- **Exploit Time:** Bundle Start
Secure Execution

- **OSGi Vulnerabilities – Characterization**
  - Freezing Management Utility – Hanging Thread

**Vulnerability Description**

- **Description:** A hanging thread in the Bundle Activator makes the management utility freeze.
- **Preconditions:** -
- **Attack Process:** -
- **Consequence Description:** Block the OSGi Management entity (the felix, equinox, or knopflerfish shell; when launched in the KF graphical interface, the shell remain available but the GUI is frozen).
- **See Also:** Management Utility Freezing - Infinite Loop, Hanging Thread
Secure Execution

- OSGi Vulnerabilities – Characterization
  - Freezing Management Utility – Hanging Thread

Protection

- Existing Mechanisms: -
- Enforcement Point: -
- Potential Mechanisms: OSGi Platform Modification - Bundle Startup Process (launch the bundle activator in a separate thread); Code static Analysis
- Attack Prevention: -
- Reaction: -
Secure Execution

- OSGi Vulnerabilities – Characterization
  - Freezing Management Utility – Hanging Thread

Vulnerability Implementation

- Code Reference: fr.inria.ares.hangingthread-0.1.jar, fr.inria.ares.hangingthread2-0.1.jar
- OSGi Profile: J2SE-1.5
- Date: 2006-08-24
- Test Coverage: 20%
- Known Vulnerable Platforms: Felix; Equinox; Knopflerfish; Concierge
- Known Robust Platforms: SFelix
Secure Execution

- OSGi Vulnerability Catalog
  - Bundle Archive
    - 3 occurrences
  - Bundle Manifest
    - 3 occurrences
  - Bundle Activator
    - 2 occurrences
  - Bundle Code - Native
    - 2 occurrences
  - Bundle Code - Java
    - 13 occurrences
- Bundle Code – OSGi API
  - 6 occurrences
- Bundle Fragment
  - 3 occurrences
Secure Execution

- OSGi Vulnerability Catalog
  - Neuman and Parker's classification

Intrusion Techniques against OSGi

- NP4 – Setting up Subsequent Misuse
- NP5 – Bypassing Intended Control
- NP6 – Active Misuse of Resources
- NP7 – Passive Misuse of Resources
- NP8 – Misuse resulting of Inaction
Secure Execution

- OSGi Vulnerability Catalog
  - Linqvist's classification

Intrusion Results in an OSGi Platform

- Denial of Service: 23
- Exposure: 8
- Erroneous Output: 8
Secure Execution

- Potential Security Mechanisms
  - Hardened Platform
    - Adapt the behavior of the Platform to prevent identified flaws
  - Execution Permission
    - Specified, hardly available
    - Give execution rights to trusted Bundle Provider only
  - Code analysis (PCC-like)
    - No memory leak, no infinite loops
Secure Execution

- Hardened OSGi
  - INRIA Sflexix Project Prototype, V0.2
  - 9 vulnerabilities out of 32 patched
  - 13 more are protected with Java Permissions
  - 69% of vulnerabilities prevented
  - Felix + permissions: 44%
  - Equinox + permissions: 53%
Secure Execution

- Hardened OSGi
Secure Execution

- **Recommendations for the OSGi Specifications**
  - Do not rely on the embedded Java Archive verifier
  - Bundle Resolution Process should be robust
    - Ignore duplicate imports
    - Handle large manifests without radical performance breakdown
  - Bundle Start Process
    - Start the Bundle Activator in a separate process
  - OSGi Service Registration
    - Explicit limitation of the number of registered services
    - Absolute Maximum could be 50?
Secure Execution

- **Recommendations for the OSGi Specifications**
  - **Bundle Installation process**
    - Maximum storage size of bundle archive (for embedded devices)
    - Should be performed before download when relevant
  - **Bundle Uninstallation process**
    - Remove Bundle data on the local file system
Secure Execution

- CBAC - Component-based Access Control
  - Java permissions are not a panacea for Components
    - Example: single RMI Call

![Java Permissions and Performances Graph]

- 1.97: Ratio SM On/SM Off
- Felix+Sun JVM, Concierge+Sun JVM, Concierge+JamVM, Felix+JamVM (S), Laptop, Linksys NSLU2 (SLUG)
Secure Execution

- CBAC
  - Java permissions are not a panacea for Components
    - Performance, runtime abortion of applications, undefined management process
    - Simply not used in the real life
Secure Execution

- CBAC
  - Validation of execution permission at install time
Secure Execution

- Comparison of Security Mechanisms for OSGi

![Graph showing the protection rate for secure OSGi platforms. The graph compares Default, Hardened OSGi, Java Permissions, Hardened + Permissions, CBAC, and Hardened + CBAC configurations. The protection rates are 0.06, 0.28, 0.41, 0.69, 0.47, and 0.75 respectively.]
Secure Execution

- 5 Challenges for Secure OSGi Platforms
  - Infinite loop in method call/hanging Thread
    - Method does not return (Java)
  - Memory Load Injection
    - If Pointers to object are kept, GC does not help (Java)
  - Decompression Bomb
    - (Java)
  - Exponential Thread Number
    - Crashes the JVM (Java)
  - Service Short Circuit
    - SOP-level vulnerability (OSGi)
Summary

- Extensible Component Platforms
- Dependability for Extensible Component Systems
- Secure Deployment
- Secure Execution

**Achievements and Open Issues**
Achievements and Open Issues

• Dependability Properties for Execution
  – For Dynamic Component Systems

<table>
<thead>
<tr>
<th></th>
<th>Fault Prevention</th>
<th>Attack Prevention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>Prevent stopping/destruction of Platforms and Components</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>Prevent Erroneous Method Calls</td>
<td></td>
</tr>
<tr>
<td>Safety</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(No full guarantee on Java-based Systems)</td>
<td></td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Ressource Isolation</td>
<td></td>
</tr>
<tr>
<td>Integrity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>Component Life-Cycle by default in OSGi</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Separation of Management and Applicative level required</td>
<td></td>
</tr>
</tbody>
</table>
Achievements and Open Issues

- OSGi Security is not yet mature
- 5 security profiles that need to be supported
  - Specifications
    - An implementation of the OSGi R4
  - Life-Cycle
    - Supporting Bundle Deployment
  - Management
    - Secure Connection to the management utilities
  - Critical Applications
    - Banking Applications
  - Multi-User Applications
    - Access Control and isolation

*Improvement brought ... a complete solution is still to be implemented*
Conclusions

● Dependability Framework for secure systems
  – An all-encompassing overview
  – Identification of research and development requirements
● Two aspects of Component Security
  – Deployment
  – Execution
● Much work is still required in order to provide secure Extensible Component Platforms
  – PCC-like Code analysis to support execution of code from unknown Providers
Questions?