Dependability for Java Mobile Code
- A pragmatic research view

Pierre Parrend
CITI Laboratory
INRIA – ARES Team
Lyon, France

OWASP Swiss Chapter
Meeting July 2007
Zürich
The Vision

• **A Net of Applications**
  - Interconnected world
    • Web Servers, Handheld Devices, Home PC, Home Boxes
    • Each device can consume and use services
  - Shifting programming model
    • Client Server webs apps are no longer satisfactory for mobile devices
    • Ressource limited devices need extensible execution environment: Mobile Java Apps (MIDP, OSGi, ...)
  - Consequence on Security
    • Specific approach to security concerns
The Vision

- Example – On-board desktop

Geo-localized Services

Remote Desktop

Friend Desktop

Secure Connection

GPS

Telematics Box

GPS
Summary

- **The OWASP and the Java World**
  - The OWASP Java Project
  - From Client-Server to extensible Applications
- **Dependability for Java Mobile Code**
- **A Contribution for Hardened OSGi Platforms**
Java and the OWASP

• The OWASP Java Project
  – Started 30 June 2006
  – Mailing List: 111 members
  – Articles: 26
  – Growing ...

• Related Development Projects
  – LAPSE
    • Lightweight Analysis for Program Security in Eclipse
    • Benjamin Livshits
Java and the OWASP

- The OWASP Java Project
  - Targeted at Web Application Servers
  - Focus on 4 questions
    - J2EE Security for Architects
    - J2EE Security for Developers
    - J2EE Security for Deployers
    - J2EE Security for Analysts and Testers
  - Work in progress
From Client-Server to Extensible Applications

• Motivation
  - Restricted applications for mobile devices

• Classical Web Client-Server Approach
  - Desktop Browser - rich user experience requires sufficient client side-resources (memory, screen size)
  - Java Applets, Web start (and many others) for Web-based applications

• Connection and Apps for Mobile Devices
  - Wap access for mobile devices
  - Default apps for mobile devices
From Client-Server to Extensible Applications

- Solution: Extensible Component Platforms for embedded devices
  - Existing technologies
    - Java MIDP, OSGi
  - Target systems
    - Mobile phones, automotive entertainment, home gateways, e-health systems
  - Features
    - Discovery of Apps Repositories
    - Installation of new Apps during runtime
    - Multi-Application systems
    - Uninstallation of Apps
From Client-Server to Extensible Applications

- Extensible Component Platforms prove to be powerful for server management too
  - Benefits
    - No reboot required
    - Centralized (and possibly remote) component management
    - Transparent update of System and Applications
  - Eclipse IDE
    - Based on OSGi Equinox
  - IBM Websphere 6.1
  - JBoss
    - OSGi Felix
From Client-Server to Extensible Applications

- Java Extensible Component Platforms
  - MIDP vs. OSGi
  - MIDP
    - CLDC (Connected Limited Device Configuration) Profile
    - Very lightweight environments
    - e.g.: Mobile Phones
  - OSGi
    - J2ME CDC (Connected Device Configuration) Foundation Profile
    - Lightweight or standard environments
    - e.g.: PDAs
From Client-Server to Extensible Applications

- MIDP
  - Mobile Information Device Profile
  - Defined by Sun
  - Applications
    - Middlet Suites
    - Defined in an external JAD File
      - Java Application Descriptor
From Client-Server to Extensible Applications

- OSGi
  - Was 'Open Service Gateway Initiative'
    - Is now an adjective
  - Forstered by the OSGi Alliance
  - The Platform - The Bundles
From Client-Server to Extensible Applications

- OSGi
  - Communication between bundles
    - Package or Services
    - Internal Description, enables Dependency Resolution
      - And thus dynamic discovery
Summary

• The OWASP and the Java World

• Dependability for Java Mobile Code
  – From Security to Dependability
  – Security for Java Mobile Code: State of the Art

• A Contribution for Hardened OSGi Platforms
From Security to Dependability

• Java Extensible Component Platforms: an Evolving Threat Model
  - Web Servers
    • Hackers can come from the Internet
    • Attack Surface is kept as small as possible
  - Extensible Component Platforms
    • Hackers can come from the Internet
    • Hackers can hide malware in Components
    • Attack Surface is as big as the Specification ...
      - Or at least is made of all actions the Component is allowed to do
From Security to Dependability

● A new approach to security is required
  – A firewall is not enough
    • AAA model outdated
  – Control on code is more necessary than ever
    • It is so easy to block a system when executing code on it
  – Current JVMs are designed for secure execution of single applications
    • Multi-Application save ressource
    • But are likely to bring big troubles
  – Dependability
    • Security + Robustness
From Security to Dependability

- Dependability

```
secrecy properties

functional properties

"simple"   "complex"

Safety  Security
```
From Security to Dependability

- Threat Model for Extensible Component Platform
  - Deployment
From Security to Dependability

- Threat Model for Extensible Component Platform
  - Execution – At the Example of the OSGi Platform
    - Each Element of the Execution Platform Can be the source of vulnerabilities
    - JVM
      - Execution Platform
      - API
    - OSGi Platform
      - Life-Cycle Layer – bundle management
      - Module Layer – package management
      - Service Layer
Security for Java Mobile Code: State of the Art

- Principle of Security for Java Code
  - Strong Data Typing
    - No buffer overflow
  - Automatic Memory Management
    - No memory leak
  - Bytecode verification
    - Before execution
  - Secure Class Loading
    - Permission mechanism
Security for Java Mobile Code: State of the Art

• MIDP Security
  – Three security levels
    • Low-level ~ virtual machine level security
    • Application-level ~ applications do not escape 'sandbox'
    • End-to-end ~ Security in all phases of e.g. a connection via e.g. encryption
  – Digital signature to enable trusted applications (only after CLDC 1.1)
    • Manufacturer, operator, trusted 3rd party, untrusted
    • Needed for phone calls, push networking features, etc
    • User authorization may also be used if the trust level is not enough for certain feature
  – Midlet Signature: in the JAD File
Security for Java Mobile Code: State of the Art

- MIDP Security

<table>
<thead>
<tr>
<th>End-to-end security:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Security in all phases of e.g. a connection via e.g. encryption</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application-level security:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Do not escape sandbox</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Low-level security:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Virtual machine level</td>
</tr>
</tbody>
</table>
Security for Java Mobile Code: State of the Art

- MIDP Security

Tommi Mikkonen, Uni. Tampere (Fi.)
Security for Java Mobile Code: State of the Art

- OSGi Security
  - Secure Deployment
• OSGi Security
  – Digital Bundle Signature
Security for Java Mobile Code: State of the Art

• OSGi Security
  – Java Permissions
  – OSGi Permissions
    • AdminPermission
      – Lifecycle, metadata, listener, execute
    • PackagePermission
      – Export, import
    • ServicePermission
      – Register, get
Security for Java Mobile Code: State of the Art

- OSGi Security
  - Permission Management
    - At runtime
  - Conditional Permissions
    - Perform additional check

```java
{
  [ ..BundleSignerCondition "* ; o=ACME" ]
  ( ..AdminPermission "(signer=\* ; o=ACME)" "*" )
  ( ..ServicePermission "..ManagedService" "register" )
  ( ..ServicePermission "..ManagedServiceFactory"
    "register"
  )
  ( ..PackagePermission "..cm" "import" )
}
```
Security for Java Mobile Code: State of the Art

• Current Security Level
  – Secure Deployment
  – Restrictions on execution are possible

• Requirements
  – No Guarantee on the executed code
    • Simply trust the Issuer
  – Research efforts
    • Proof Carrying Code
      – Can only prove subsets of programming languages
      – 'I can tell you that your virus will never crash', Peter Lee
Summary

- The OWASP and the Java World
- Dependability for Java Mobile Code
- **A Contribution for Hardened OSGi Platforms**
  - Engineering Dependable Applications
  - Toward a Hardened OSGi Platform
Engineering Dependable Applications

• Requirement
  - Life-Cycle long support of security

• The Bundle Life-Cycle

![Diagram showing the bundle life-cycle phases]

- Development
- Digital Signature
- Discovery
- Download
- Installation
- Execution

- Packaging
- Meta-data generation
- Publication

- Developer's PC
- Publication Tool
  - SF-Jarsigner
- Bundle Repository Server
- OSGi Client Platform
Engineering Dependable Applications

- Secure Coding throughout Bundle Life-Cycle
Engineering Dependable Applications

- Tools for Secure Deployment of OSGi Bundle
Engineering Dependable Applications

- Sfelix
  - [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
    - In a near future – still need to be published
Engineering Dependable Applications

- Sfelix

```java
-> obr start "HTTP Service"
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:1967)
    at org.apache.felix.framework.Felix.installBundle(Felix.java:1922)
    at org.apache.felix.framework.BundleContextImpl.installBundle(BundleContextImpl.java:90)
    at org.apache.felix.framework.BundleRepositoryImpl.createServer(BundleRepositoryImpl.java:257)
    at org.apache.felix.framework.BundleRepositoryImpl.createServer(BundleRepositoryImpl.java:356)
    at org.apache.felix.framework.BundleRepositoryImpl.createServer(BundleRepositoryImpl.java:294)
    at org.apache.felix.framework.BundleRepositoryImpl.createServer(BundleRepositoryImpl.java:108)
    at org.apache.felix.shell.impl.Activator$ShellServiceImpl.executesCommand(Activator.java:253)
    at org.apache.felix.shell.shell.ShellShell<ScriptHandle<run(Activator.java:165)
    at java.lang.Thread.run(Thread.java:655)

Caused by: org.osgi.framework.BundleException: Bundle: Unsecure
    at fr.inria.arches.framework.cache.DefaultSecuredBundleArchive.checkArchiveValidity(DefaultSecuredBundleArchive.java:73)
    at org.apache.felix.framework.Felix.installBundle(Felix.java:1923)
    ...
```

Engineering Dependable Applications

- The SF-JarSigner Tool
  - http://sf-jarsigner.gforge.inria.fr/
  - The Archive Analysis Panel
Engineering Dependable Applications

- The SF-JarSigner Tool
  - The Bundle Repository Management Panel
Engineering Dependable Applications

- The SF-JarSigner Tool
  - The Bundle Publication Panel
Toward a Hardened OSGi Platform

- Requirements
  - Specification for an hardened OSGi platform

- OSGi Platform Model

![Diagram showing service management, life-cycle management, dependency resolver, and execution components.]

- Component Repository
- Component Downloader
- Local execution
- Local interactions
- Service Management
- Life-Cycle Management
- Dependency Resolver
- Execution

deploy
Toward a Hardened OSGi Platform

- 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
Toward a Hardened OSGi Platform

- Specific Taxonomies for the OSGi extensible Component Platform

![Diagram](image)

- Dependability Models for component platforms
  - Vulnerability Source
  - Location of Exploit Code
  - Attack Targets
  - Attack Consequences

- OSGi Life-Cycle Layer
- OSGi Bundle
- OSGi Platform
- Crash

- Models
- Examples
Toward a Hardened OSGi Platform

- Building a robust OSGi Platform
  - Identified Protection Mechanisms
    - Platform hardening
    - Java Permissions
    - Code Analysis
  - Hardened OSGi Platform
    - INRIA Sfelix Project Prototype, V0.2
    - http://sfelix.gforge.inria.fr/
    - 8 vulnerabilities out of 29 patched
    - 13 more are protected with Java Permissions
    - 75% of vulnerabilities prevented
    - Felix: 48%
    - Equinox: 58%
Toward a Hardened OSGi Platform

• Recommendations for the OSGi Specifications
  – Do not rely on the embedded Java Archive verifier
    • OSGi R4, Paragraph 2.3
  – Bundle Resolution Process should be robust
    • Ignore duplicate imports (currently: abort; see R4 par. 3.5.4; Equinox ignores)
    • Handle large manifests without radical performance breakdown
  – Bundle Start Process
    • Start the Bundle Activator in a separate process (R4 par. 4.3.5)
  – OSGi Service Registration
    • Explicit limitation of the number of registered services (R4 par. 5.2.3)
    • Absolute Maximum could be 50?
Toward a Hardened OSGi Platform

- **Recommandations for the OSGi Specifications**
  - **Bundle Installation process**
    - Maximum storage size of bundle archive (for embedded devices) (R4 par. 4.3.3)
    - Should be performed before download when relevant
  - **Bundle Uninstallation process**
    - Remove Bundle data on the local file system (R4 par. 4.3.8)
Conclusions

• Java Mobile Apps are taking off
  – OWASP is active in the applicative domain too
  – Shift from Security to Dependability focus

• Need of a Life-Cycle long control
  – Security keeps being a management-level question

• OSGi is one solution
  – With so far only reduced implemented security features
Questions?