Apps in Cars: JamaicaCAR and how it ensures security

Dr. Fridtjof Siebert, CTO
aicas GmbH
Mobile Forum Stuttgart
17 May 2013
The aicas group

- Embedded and Realtime Java
- Automotive App Framework
- Analysis and Debugging Tools for Java

- **Head Office**: Karlsruhe, Germany
- **Sales Office**: Paris, France
- **aicas inc.:** Stamford, Connecticut
Embedded Realtime Applications

Markets:

- industrial automation
- avionic/satellite
- automotive
- medical
Embedded Realtime Applications

Markets:
- industrial automation
- avionic/satellite
- automotive
- medical

Customers:
- BOEING
- CLAAS
- SIEMENS
- Honeywell
- CASSIDIAN
- øerlikon
- KAI
- MULTIVAC
- USTER
- Berner & Mattner
- EADS
- Harman International
In-car App Examples
Apps in CARs

- Additional Concerns
  - Applications running on Infotainment System are getting more and more safety relevant
  - Must not distract the driver
  - Important Apps must not be affected by less important ones
  - Driver Assistants are becoming directly safety relevant
Apps in JamaicaCAR

• An App is a JAR archive containing
  - Collection of Java class files
  - One main class that extends Xlet interface
  - Resource files (images, etc.)
  - Security related files
    • permissions
    • resource budgets
Apps are Xlets

Interface
Xlet

initXlet(XletContext ctx);
startXlet();
pauseXlet();
destroyXlet();
App Life Cycle

install → Stopped
uninstall → Stopped

Stopped → Running
Stopped → Paused

Running → Stopped
Running ↔ Paused

Paused ↔ Running
Paused ↔ Stopped

initXlet() → Running
initXlet() → Paused

pauseXlet() → Stopped
pauseXlet() → Paused

destroyXlet() → Stopped
destroyXlet() → Paused

startXlet() → Running
startXlet() → Paused
App Signing Process

Xlet Developer writes Xlet JAR

Platform Provider or Trusted Entity:
add budgets and permissions, checks and signs JAR

Private Key

Xlet Distributor

Xlet JAR + signature

Target Device:
checks signature runs application enforces budgets and permissions

Public Key
Xlet checking before Signing

• **Classfile Verification**
  - off-load the task of classfile verification from the target device

• **Check Permitted APIs**
  - analyse code to use only APIs it is allowed to use
  - no arbitrary library calls permitted
**Xlet Permissions**

- Permissions grant rights

```java
grant {
}

grant {
    permission java.io.FilePermission "/storage/data.txt", "read";
}
```

- SecurityManager checks permissions at runtime
Xlet Budgets

• **Budgets specify resource limits**

```javascript
xlet.maxNumberOfThreads = 20
xlet.period = 100ms
xlet.timeBudget.paused = 10ms
xlet.timeBudget.running = 80ms
```

[...]
App Framework Architecture

- LWUIT / OpenGL-ES Graphics
- Network Connectivity (Apache Client)
- Application Management Services

JamaicaVM
Partitioning

- **Space Partitioning**
  - using Java mechanisms and realtime GC

- **Time Partitioning**
  - using CPU monitoring in JVM scheduler
Space Partitioning

- **Java Strict Typing**
  - no pointer arithmetic etc.

- **Distinct Class Loaders**
  - provide name space local to Xlet

- **Enhanced Java Libraries**
  - no static fields to exchange data between Xlets

- **Inter-Xlet Communication via Serialization**

- **Realtime GC**
Memory Garbage Collection

• **Realtime GC**
  - forbids accessing memory in other Xlets
  - frees unused memory automatically
  - enabled control of memory usage per Xlet
  - enables charging GC CPU usage to Xlets
Classic Garbage Collector

Thread:

GC
User 1
User 2

time

GC may stop execution for long periods of time
- long, unpredictable pauses
Realtime Garbage Collector

Thread: rt1, rt2, rt3, rt4, ...

time

All Java Threads are realtime threads

- GC-work performed at allocation time only
- WCET for memory allocation
- GC exact and deterministic
Realtime Garbage Collector for Xlets

Advantages of using realtime GC

• Xlets “pay” for allocation by doing GC work
• no GC pauses
• GC work fully preemptible
  - Xlets unaffected by allocation in other Xlets
• per-Xlet memory usage monitoring
• per-Xlet OutOfMemoryError
  - if memory is low or Xlet exceeds budgets
Time Partitioning

- **Per Xlet CPU budgets**
  - CPU usage monitoring for all threads
  - Checks at regular Intervals
  - Counter measures
    - Priority reduction
    - forced Xlet termination
Terminating Errant Xlets

- **Unresponsive Xlets may be “killed”**
  - Xlets stuck in endless loop
  - exceeding memory budget
  - not returning from `destroyXlet()` call

- **Forced termination possible**
  - stopping threads by VM mechanisms
  - closing all resources (files, sockets, etc.)
  - freeing memory using realtime GC
Conclusion

JamaicaCAR security mechanisms
- ensure code is not tampered with
- enforce resource budgets
- enforce access permissions
- protect system from errant Apps
- lifecycle: launch, hide, terminate applications
- forced termination
Apps in Cars: JamaicaCAR and how it ensures security

Dr. Fridtjof Siebert, CTO
aicas GmbH
Mobile Forum Stuttgart
17 May 2013