Avionics Systems Hosted on a distributed modular electronics Large scale dEmonstrator for multiple tYpe of aircraft

Presented by
Walter Binder (Università della Svizzera italiana, Switzerland)

Prepared by
Walter Binder, Lubomir Bulej, Haiyang Sun, Yudi Zheng (Università della Svizzera italiana, Switzerland)

Comprehensive Multi-Platform Dynamic Program Analysis for the Java and Dalvik Virtual Machines

ASHLEY-WP45-3-2-DISM-PRES-01-0481

This document is produced under the Grant Agreement 605442. It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee. “Unrestricted PUBLIC Access”
Preface

This publication only reflects the view of the ASHLEY Consortium or selected participants thereof. Whilst the ASHLEY Consortium has taken steps to ensure that this information is accurate, it may be out of date or incomplete, therefore, neither the ASHLEY Consortium participants nor the European Community are liable for any use that may be made of the information contained herein.

This document is published in the interest of the exchange of information and it may be copied in whole or in part providing that this disclaimer is included in every reproduction or part thereof as some of the technologies and concepts predicted in this document may be subject to protection by patent, design right or other application for protection, and all the rights of the owners are reserved.

The information contained in this document may not be modified or used for any commercial purpose without prior written permission of the owners and any request for such additional permissions should be addressed to the ASHLEY co-ordinator (Thales Avionics S.A., 105 Av. du General Eisenhower, BP 63647, 31036 Toulouse, FRANCE, for the attention of the ASHLEY Project Manager) in the first instance.

This document is produced under the Grant Agreement 605442. It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee.

“Unrestricted PUBLIC Access”
Dynamic Program Analysis in Software Development

- Code
  - Contains
  - Bugs / Defects
  - Improve
- Fix / Optimization
  - Propose
  - Analysis
  - Employ

This document is produced under the Grant Agreement 605442. It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee. “Unrestricted PUBLIC Access”
A Plethora of Tools

jdb - The Java Debugger

Program Analysis

Debugging

VisualVM 1.3.8
All-in-One Java Troubleshooting Tool

BTRACE

NetBeans

eclipse

LOG4J

Memory Analyzer (MAT) is found in the
Eclipse IDE

chronon

DVR for Java

Cobertura

Testing

Patty (Java Performance Analysis Tool)

Profiler4j

Profiling made easy

ORACLE

JROCKIT

JIP — The Java Interactive Profiler

JIPprofiler

This document is produced under the Grant Agreement 605442.
It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee.

“Unrestricted PUBLIC Access”
Our Goal

All-in-one Dynamic Analysis Framework

- **Code**
- **Testing**
  - Code coverage
  - Crash reproduction
  - ...
- **Debugging**
  - Memory leak detection
  - Data race detection
  - ...
- **Tuning**
  - Calling context profiling
  - Object lifetime profiling
  - ...

Improved Code

Unrestricted PUBLIC Access
Instrumentation-based Dynamic Program Analysis

Bytecode

Instrumentation

Analysis

Event Producer

Event Consumer

This document is produced under the Grant Agreement 605442. It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee.

“Unrestricted PUBLIC Access”
Bytecode Instrumentation

Ease of use

Flexibility

AspectJ

ASM

This document is produced under the Grant Agreement 605442. It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee. “Unrestricted PUBLIC Access”
Design Goals

- High-level programming model
  - Specification language for capturing events

- Expressiveness
  - Any instrumentation can be expressed

- Coverage
  - From VM start until termination

- Isolation
  - Avoid interference with observed program
DiSL

- **DSL for Instrumentation**
  - Embedded in Java

- Aspect-oriented features
  - What You See Is What You Instrument!

- Designed for observation
  - Inserted code cannot change state of observed program

- Built on top of **ASM**

Marek et al., DiSL: A Domain-specific Language for Bytecode Instrumentation. AOSD’12
Shadow VM

- Asynchronous remote execution of analysis
  - Isolation

- Shadow programming model
  - Attach shadow state easily

Marek et al., ShadowVM: Robust and Comprehensive Dynamic Program Analysis for the Java Platform. GPCE’13
Dynamic Analysis as an Event-processing System

Event Producer

Native Agent

JNI Calls

Socket

Event Consumer

Observed VM

Shadow VM

Unrestricted PUBLIC Access
Shadow VM Architecture (1)

Instrumentation Server
- User-supplied instrumentation code
- Instrumentation framework (+ host JVM)

Instrumented base program
- Instrumented class library
- Native agent
- Event API

Analysis Server
- User-supplied analysis code
- Shadow API implementation (+ host JVM)

Instrumentation process
- Instrumentation request

Analysis process
- Event
Shadow VM Architecture (2)

- **Instrumentation Server**
  - User-supplied instrumentation code
  - Instrumentation framework (+ host JVM)

- **Analysis Server**
  - User-supplied analysis code
  - Shadow API implementation (+ host JVM)

- **Observed VM**
  - Instrumentation request
  - Event

- **Instrumentation process**
  - Server

- **Analysis process**
  - Server

This document is produced under the Grant Agreement 605442. It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee. “Unrestricted PUBLIC Access”
Event Types

Event Producer

Library Event
User Event
Lifecycle Event
IPC Event

Native Agent

Observed VM

Event Consumer

Shadow VM
Shadow API (1)

Event Producer

★

- Object
- Class
- int

Native Agent

Event Consumer

- ShadowObject
- ShadowClass
- int

Observed VM

Shadow VM
Shadow API (2)

```
ShadowObject
+ getId ():long
+ getState ():Object
+ setState (state :Object):void
+ getShadowClass ():ShadowClass

ShadowString
+ toString ():String

ShadowThread
+ getName ():String
+ isDaemon ():boolean

ShadowClass
+ getName ():String
  ... type information methods ...
+ getSuperclass ():ShadowClass
+ getClassLoader ():ShadowObject
```
Shadow API (3)

Event Producer

Event Consumer

- ShadowObject
- ShadowClass
- int

Native Agent

Observed VM

Shadow VM

This document is produced under the Grant Agreement 605442.
It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee.

“Unrestricted PUBLIC Access”
Multi-process Support

Event Consumer

Shadow VM

Shadow Spaces

This document is produced under the Grant Agreement 605442. It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee.

“Unrestricted PUBLIC Access”
Android Support

- Q: Changes to existing analyses?
  A: No (the good)

- Q: Different bytecode format?
  A: We transform to Java bytecode (the bad)

- Q: Missing DVMTI?
  A: We modify DVM (the ugly)

Sun et al., A Programming Model and Framework for Comprehensive Dynamic Analysis on Android. Modularity’15
Tools

- Multi-platform
  - Calling context profiler
  - Code coverage
  - Object lifetime
  - Field immutability
  - …

- Android-specific
  - Permission usage analyzer
  - Network communication tracker
### Example Tools: Implementation Effort

- **JaCoCo**: code coverage analysis
- **ElephantTracks**: object-lifetime analysis

<table>
<thead>
<tr>
<th></th>
<th>JVM</th>
<th>DVM</th>
<th>Full Coverage</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Producer</td>
</tr>
<tr>
<td>Original JaCoCo</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>1389</td>
</tr>
<tr>
<td>ShadowVM JaCoCo</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>281</td>
</tr>
<tr>
<td>Original ElephantTracks</td>
<td>Only Java 1.6</td>
<td>No</td>
<td>Yes</td>
<td>6668</td>
</tr>
<tr>
<td>ShadowVM ElephantTracks</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>608</td>
</tr>
</tbody>
</table>

Zheng et al., Comprehensive Multi-platform Dynamic Program Analysis for Java and Android.

*To appear in IEEE Software*
Permission usage tracking

- Uses IPC events to track permission checks triggered by apps

<table>
<thead>
<tr>
<th>process</th>
<th>thread id</th>
<th>permissions</th>
<th>call stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>pixelRush.xphonefree(2016)</td>
<td>11</td>
<td>android.permission.READ_CALL_LOG</td>
<td>open</td>
</tr>
<tr>
<td>pixelRush.xphonefree(2016)</td>
<td>12</td>
<td>android.permission.READ_CALL_LOG</td>
<td>open</td>
</tr>
<tr>
<td>pixelRush.xphonefree(2016)</td>
<td>1</td>
<td>android.permission.ACCESS_ALL_EXTERNAL_STORAGE</td>
<td>open</td>
</tr>
<tr>
<td>pixelRush.xphonefree(2016)</td>
<td>1</td>
<td>android.permission.ACCESS_ALL_EXTERNAL_STORAGE</td>
<td>open</td>
</tr>
<tr>
<td>pixelRush.xphonefree(2016)</td>
<td>12</td>
<td>android.permission.READ_PHONE_STATE</td>
<td>open</td>
</tr>
<tr>
<td>pixelRush.xphonefree(2016)</td>
<td>16</td>
<td>android.permission.READ_PHONE_STATE</td>
<td>open</td>
</tr>
<tr>
<td>pixelRush.xphonefree(2016)</td>
<td>16</td>
<td>android.permission.GET_ACCOUNTS</td>
<td>open</td>
</tr>
</tbody>
</table>

This document is produced under the Grant Agreement 605442. It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee. “Unrestricted PUBLIC Access”
Example: Android-specific Analyses (2)

- Network communication tracking
  - Intercepts connections and data exchange

```
<table>
<thead>
<tr>
<th>proc</th>
<th>address</th>
<th>data</th>
<th>call stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>50F</td>
<td>0.0.0.0 : 0</td>
<td>N/A</td>
<td>open</td>
</tr>
<tr>
<td>50F</td>
<td>54.215.201.125 : 443</td>
<td>N/A</td>
<td>open</td>
</tr>
<tr>
<td>50F</td>
<td>123.45.67.89 : 80</td>
<td>274 bytes</td>
<td>open</td>
</tr>
</tbody>
</table>
```

```
c/a/a/cr.dolnInBackground
c/a/a/cr.a
```

This document is produced under the Grant Agreement 605442.
It is the property of the ASHLEY consortium and shall not be distributed or reproduced without the formal approval of the ASHLEY Steering Committee.

“Unrestricted PUBLIC Access”
Publications

- Marek et al., DiSL: A Domain-specific Language for Bytecode Instrumentation. *AOSD’12*
- Kell et al., The JVM is not Observable Enough (and What To Do About It). *VMIL’12*
- Marek et al., ShadowVM: Robust and Comprehensive Dynamic Program Analysis for the Java Platform. *GPCE’13*
- Ansaloni et al., Enabling Modularity and Re-use in Dynamic Program Analysis Tools for the Java Virtual Machine. *ECOOP’13*
- Sun et al., A Programming Model and Framework for Comprehensive Dynamic Analysis on Android. *MODULARITY’15*
- Zheng et al., Comprehensive Multi-platform Dynamic Program Analysis for Java and Android. *To appear in IEEE Software*
- Zheng et al., Accurate Profiling in the Presence of Dynamic Compilation. *OOPSLA’15*
Contacts

Walter Binder
walter.binder@usi.ch
Avionics Systems Hosted on a distributed modular electronics Large scale dEmonstrator for multiple tYpe of aircraft

Call identifier: FP7-AAT-2013-RTD-1
Project co-funded by the European Commission within the Seventh Framework Programme (2013-2017)