УДК 004.435, 004.423, 004.4'41

МОДИФИКАЦИЯ СКОМПИЛИРОВАННЫХ ПРИЛОЖЕНИЙ ДЛЯ ПЛАТФОРМЫ ANDROID МЕТОДОМ АСПЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Г. Ю. Зайцев, А. И. Потапкин, Д. А. Стефанцов

Описывается инструмент для модификации скомпилированных приложений для платформы Android, разработанный авторами. Инструмент основан на методе аспектно-ориентированного программирования, реализован при помощи библиотеки ASMDEX и объединяет программу приложения и аспекты за два прохода по тексту программы. Аспекты реализуются на языке Java при помощи специальных аннотаций, инкапсулирующих необходимую метаинформацию.

Ключевые слова: *АОП, Android, Dalvik.*

Аспектно-ориентированное программирование (АОП) — это способ изменения функциональности программы без изменения исходных текстов последней [1, 2]. Дополнительная функциональность при этом содержится в специальных модулях — аспектах. Удобными для реализации в виде аспектов считаются, например, требования политики безопасности, синхронизация потоков и журналирование [3].

В настоящее время распространены инструменты АОП, предполагающие, что им доступны исходные тексты программы [4, 5]. Однако зачастую требуется изменить функциональность уже скомпилированных программ. В работе описывается разработанный авторами инструмент AspectA (от слов Aspect и Android), позволяющий модифицировать приложения, скомпилированные для платформы Android [6] — одной из самых популярных мобильных платформ на сегодняшний день.

Приложения для платформы Android разрабатываются одним из трёх способов: 1) на языке программирования (ЯП) Java с последующей трансляцией в команды виртуальной машины (ВМ) Dalvik [7]; 2) на любом ЯП, допускающем трансляцию в инструкции для аппаратной архитектуры, на которой работает платформа Android; 3) совмещением двух описанных способов. Распространяются приложения в виде файлов с расширением APK, представляющих собой архив с конфигурационными файлами, файлами с машинными инструкциями для аппаратной архитектуры, файлом с расширением DEX с инструкциями для ВМ Dalvik и другими. Текущая версия AspectA написана на ЯП Java, состоит из трёх компонент (скрипта сборки `inject.sh`, утилиты для объединения DEX-файлов `DexMerger.jar` и утилиты для внедрения вызовов процедур `weaver.jar`) и работает только с приложениями, скомпилированными в инструкции для ВМ.

Интеграция исходного приложения и аспектов осуществляется скриптом `inject.sh` следующим образом: 1) с помощью утилиты `apktool` [8] распаковывается исходный APK-файл; 2) с помощью `DexMerger.jar` DEX-файлы исходного приложения и аспектов объединяются в один; 3) с помощью `weaver.jar` в часть нового DEX-файла, соответствующую исходному приложению, помещаются вызовы процедур из части нового DEX-файла, соответствующей аспектам; 4) новый DEX-файл и старые ресурсы запаковываются в новый APK-файл с помощью утилиты `apktool`. На рис. 1 показан порядок изменения файлов при внедрении аспектов в приложение.

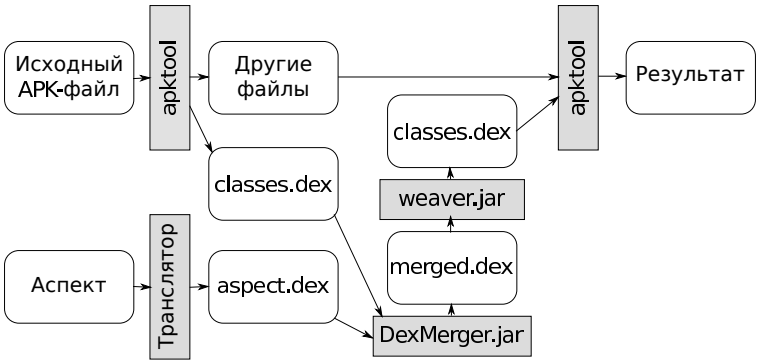

Рис. 1. Общая схема работы AspectA

AspectA реализован с помощью библиотеки ASMDEX [9], позволяющей работать с программой в инструкциях для ВМ Dalvik. Основную функциональность выполняет компонент `weaver.jar`. Для соединения приложения и аспектов байт-код приложения просматривается в два прохода: во время первого осуществляется поиск областей для внедрения процедур из аспектов, во время второго — указанное внедрение. Во время прохода библиотека ASMDEX просматривает инструкции ВМ, запуская процедуры-обработчики из `weaver.jar`. Последние анализируют текущую инструкцию и определяют её соответствие описаниям точек выполнения в аспекте. В случае совпадения в анализируемую последовательность инструкций ВМ помещаются инструкции для вызова предписания из аспекта. Использование двух проходов обусловлено необходимостью сохранения и восстановления регистров ВМ до и после вызовов внедрённых процедур соответственно. Количество дополнительных регистров, запрашиваемых у ВМ Dalvik для выполнения этого действия, невозможно вычислить на первом проходе.

Аспекты в описываемом методе должны быть реализованы на языке Java с использованием специальных аннотаций, которые добавляют к конструкциям языка необходимую метаинформацию. Альтернативами к такому подходу могли бы стать адаптация языка AspectJ [4] для работы с байт-кодом Dalvik и разработка собственного аспектно-ориентированного языка. Первый подход слишком трудоёмок по сравнению с использованным решением и, к тому же, обладает всеми ограничениями AspectJ [2]. Второй подход так же трудоёмок и лишает разработчика уже имеющихся инструментов Eclipse IDE [10], работающих для языков Java и AspectJ. В качестве примера на листинге 1 представлен аспект, который осуществляет журналирование всех вызовов член-функций, имена которых начинаются со строки `get`, при этом запись в журнал добавляется перед соответствующим вызовом член-функции (строки 2–8).

```
1 public class TestAspect extends Aspect {
2   public boolean check(JoinPoint jp) {
3     return jp.methodName.startsWith("get");
4   }
5   @BeforeExecution
6   public void advice(JoinPoint jp) {
7     Log.i("invoking method ", jp.methodName);
8   }
9 }
```

Листинг 1. Пример аспекта для AspectA

Разработанный инструмент AspectA может применяться для отладки и трассировки скомпилированных приложений для Android во время их разработки и поддержки, при этом разработчики получают возможность легко удалить всю отладочную часть программы. Полезными примерами использования данного инструмента являются также динамический анализ вредоносных приложений и реализация дополнительных механизмов защиты в уже имеющихся приложениях.

ЛИТЕРАТУРА

1. *Filman R. E. and Friedman D. P.* Aspect-oriented programming is quantification and obliviousness [Электронный ресурс] // Technical report, RIACS, 2000. URL: `http://www.riacs.edu/research/technical_reports/TR_pdf/TR_01.12.pdf`, свободный доступ (дата обращения: 9.04.2010).
2. *Стефанцов Д. А.* Реализация политик безопасности в компьютерных системах с помощью аспектно-ориентированного программирования // Прикладная дискретная математика. 2008. № 1(1). С. 94–100.
3. *Laddad R.* AspectJ in Action: Enterprise AOP with Spring Applications, 2nd edition. Greenwich, CT, USA: Manning Publications Co., 2009. 568 p.
4. `http://eclipse.org/aspectj/` — The AspectJ Project. 2013.
5. `https://sites.google.com/a/gapp.msrg.utoronto.ca/aspectc/` — Welcome to ACC: The AspeCt-oriented C compiler. 2010.
6. `http://www.android.com/about/` — Discover Android. 2013.
7. `http://code.google.com/p/dalvik/` — Dalvik. Code and documentation from Android's VM team. 2011.
8. `https://code.google.com/p/android-apktool/` — Android-Apktool. A tool for reverse engineering Android apk files. 2013.
9. `http://asm.ow2.org/asmdex-index.html` — OW2 Consortium. ASMDEX. 2012.
10. `http://eclipse.org/` — Eclipse. The Eclipse Foundation open source community website. 2013.

УДК 004.94

РАЗРАБОТКА И РЕАЛИЗАЦИЯ МАНДАТНЫХ МЕХАНИЗМОВ УПРАВЛЕНИЯ ДОСТУПОМ В СУБД MYSQL

Д. Н. Колегов, Н. О. Ткаченко, Д. В. Чернов

Работа посвящена разработке и реализации механизмов мандатного управления доступом в изначально дискреционной системе управления базами данных MySQL с использованием формальной модели безопасности. Рассматривается реализация мандатной политики управления доступом типа multilevel security (MLS). Предлагаемый мандатный механизм реализован в составе монитора безопасности ядра MySQL и позволяет выполнить основные требования обеспечения безопасности информационных потоков, предъявляемые к защищённым компьютерным системам. Ключевыми особенностями предлагаемого подхода являются формальное моделирование политики управления доступом на основе аппарата ДП-моделей, реализация мандатного управления доступом на уровне ядра СУБД, а также обеспечение требований безопасности информационных потоков.

Ключевые слова: *компьютерная безопасность, управление доступом, информационные потоки, формальные модели безопасности.*