Decompiling Android Applications

Damien Octeau, William Enck and Patrick McDaniel

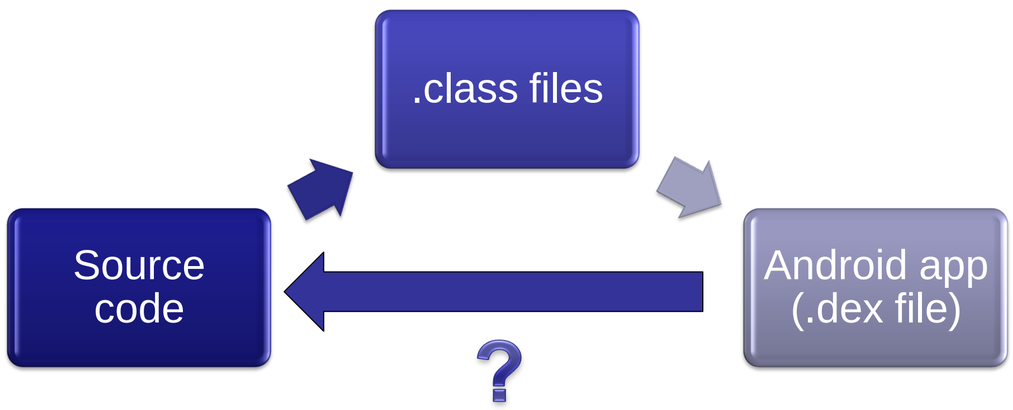
Android

➢Google's operating system for smartphones.

➢Third party applications are developed in Java but run in the Dalvik Virtual Machine, a specific VM with bytecode different from traditional Java bytecode.

➢Bytecode analysis tools already exist for Java applications (static program analysis, decompilation, etc.), but not for Android applications; in particular, there is no decompiler yet.

➢**Our solution: leverage existing tools by converting Android bytecode to traditional Java bytecode. In particular, decompilation is possible.**



The Dalvik Virtual Machine

Android applications are developed in Java and run in the Dalvik Virtual Machine, which is different from the traditional Java Virtual Machine. Those differences make the decompilation process hard.

➢Different opcodes (no trivial mapping between the two sets of opcodes).

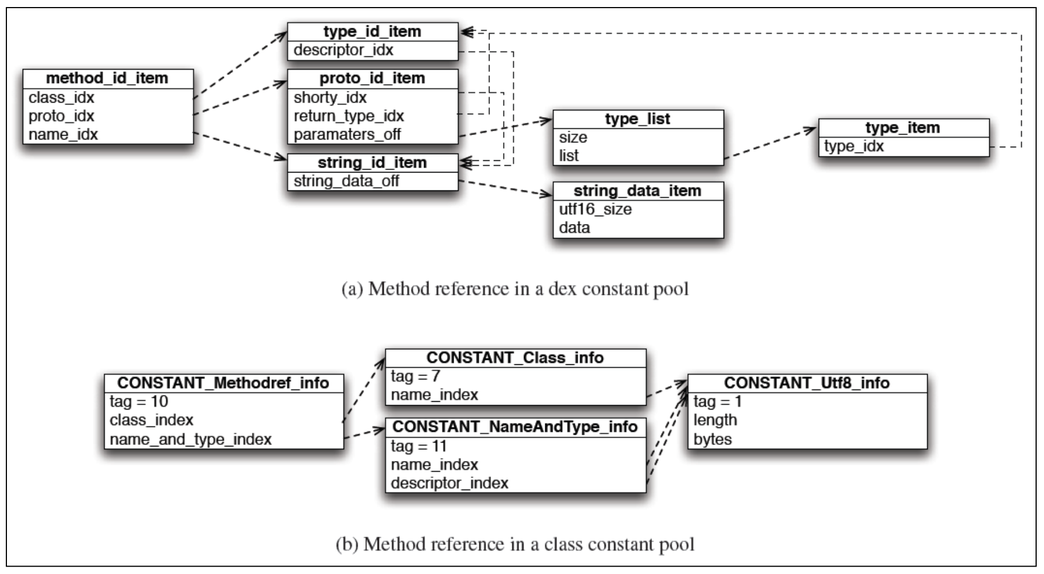
➢The Dalvik VM is register-based whereas the Java VM is stack-based.

➢Control flow compiled differently; some code restructuring is needed.

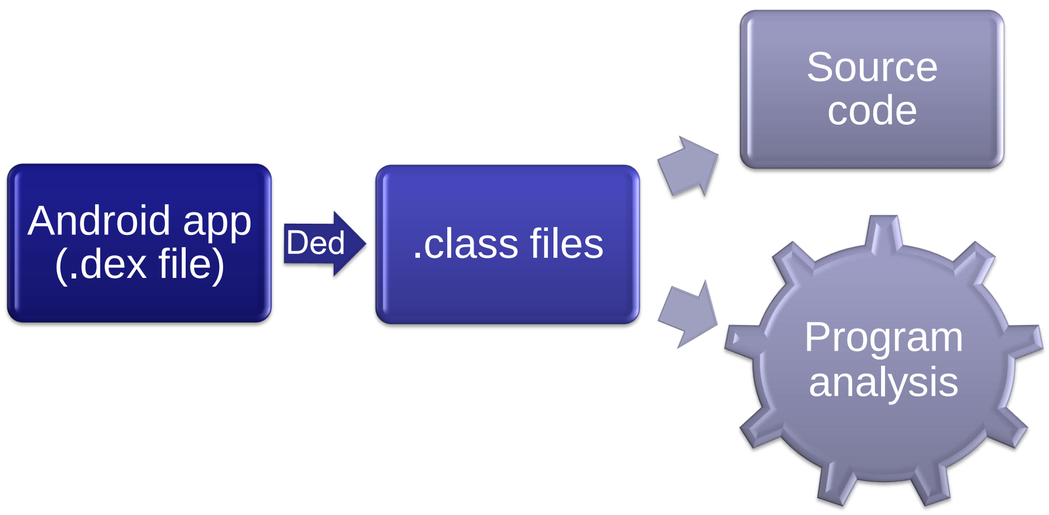
➢Not entirely type-safe; the lost type information must be recovered.

➢The Java constant pool describes more types of constants.

➢Completely different constant pool structure.



The decompilation process



➢We have created Ded, a tool which decompiles a .dex file into the corresponding .class files.

- ➢Rebuilds the constant pools
- ➢Translates over 210 opcodes
- ➢Infers types where needed

➢Our method also uses Soot, a framework for the optimization and analysis of Java bytecode, in order to get decompiled files closer to the original source code.

Initial results and future work

➢Our method enables the recovery of the source code in a very readable form.

➢This allows us to make observations about the security policies of an application.

➢We can also process the generated .class files with existing Java program analysis tools.

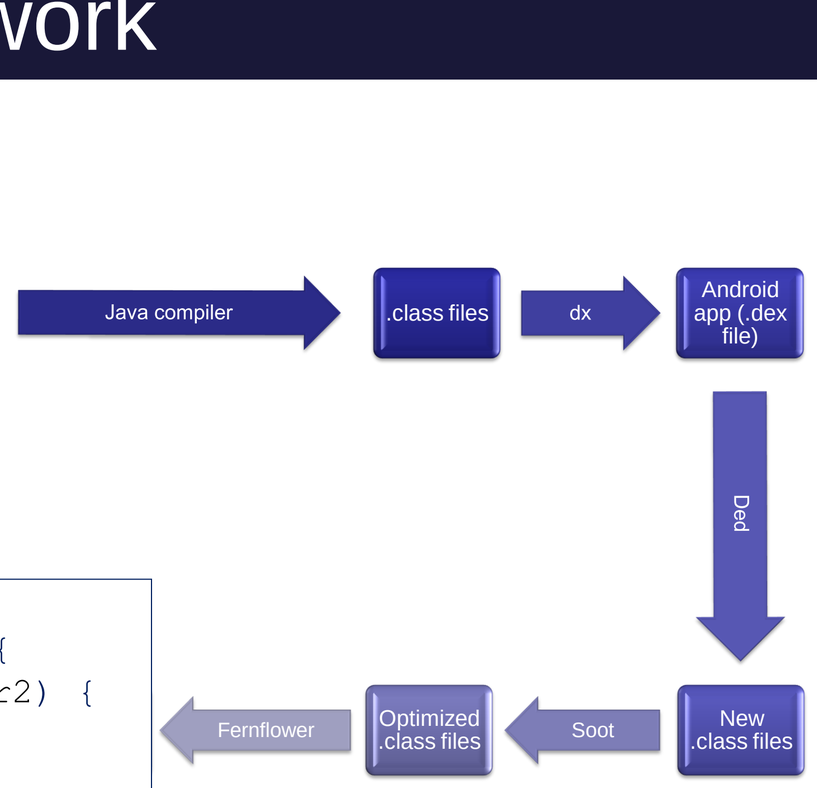
➢Future work includes refining our type recovery algorithms, as well as improving the translation of exception handlers.

Original source code:

```
public void clearTiles() {
    for (int x = 0; x < mXTileCount; x++) {
        for (int y = 0; y < mYTileCount; y++) {
            setTile(0, x, y);
        }
    }
}
```

Decompiled method:

```
public void clearTiles() {
    for(int var1 = 0; var1 < mXTileCount; ++var1) {
        for(int var2 = 0; var2 < mYTileCount; ++var2) {
            this.setTile(0, var1, var2);
        }
    }
}
```



Sponsored by NSF