Detecting Obfuscation on Android Apps

Hugo Gonzalez, Ye Ning, Yan Li, Ratinder Kaur, Natalia Stakhanova

Canadian Institute for Cybersecurity (CIC), University of new Brunswick (UNB)

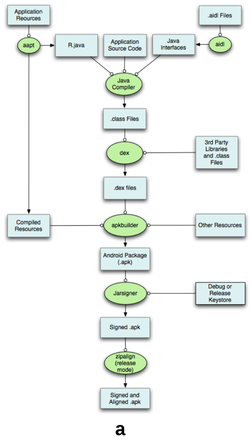
ABSTRACT

Android popularity has motivated malware authors to apply code obfuscation techniques to evade anti-malware engines. Mobile malware obfuscation can range from a simple technique that includes renaming and removal of unused identifiers to more advance techniques that includes insertion of junk code. This research work aims at detecting obfuscation tools and techniques used by malware authors. The proposed method employs a layered approach where each layer is designed to detect a particular obfuscation technique and flags the most probable obfuscation tool utilized.

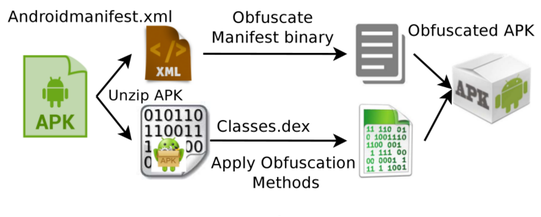
INTRODUCTION

♦ Obfuscation technique is defined as all strategies that change the content of the .dex file and/or .xml files, preserving the original functionalities of the application with or without modifying the semantic.

♦ Some obfuscators operate directly on the source files transforming them before compilation, some operate on the Java bytecode, and others operate on the Dalvik bytecode.



a



b

Figure1: (a) Android Compilation Process (b) Android App Obfuscation

OBFUSCATION TECHNIQUES

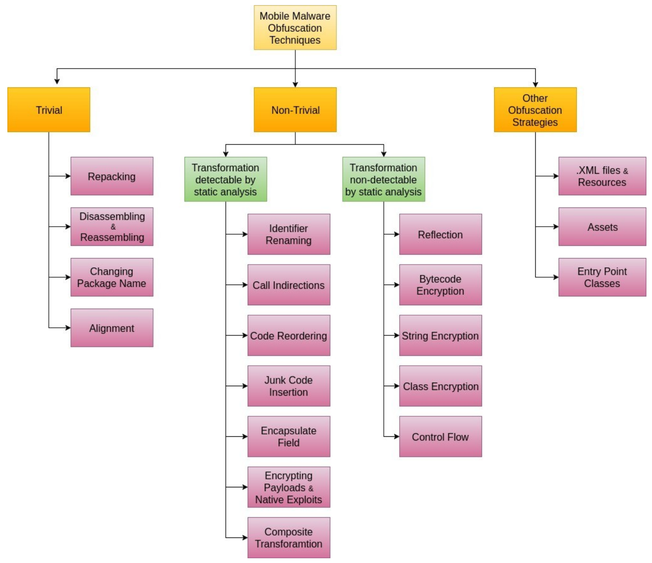
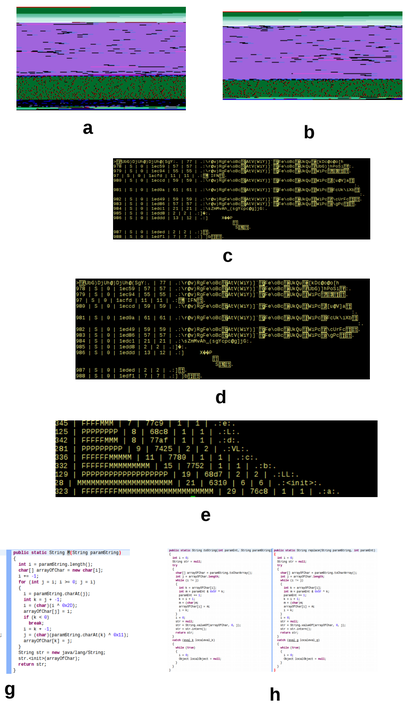


Figure3: Obfuscation Techniques



a b c d e

f g h

Figure4: (a) With optimization-black section (b) Without optimization (c) Strings Allatori (d) Encrypted strings Allatori (e) ProGuard strings (f) Shield4j mapping (g) Allatori encryption/decryption routine (h) Dasho encryption/decryption routine

PROPOSED METHOD

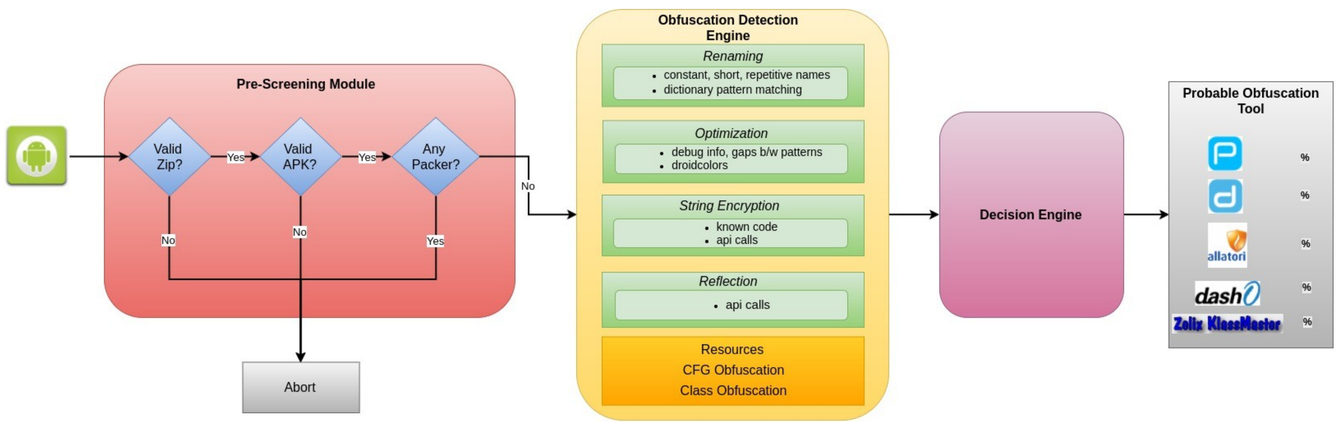


Figure2: Android App Obfuscation Detection

DATASET

♦ Android apps obfuscated with each different obfuscation option of a tool.

♦ 9 obfuscation tools, nearly 2000 apps from F-droid

Table 1: Obfuscation Tools Features

Features → / Obfuscator ↓	Obfuscate Names	Obfuscate Resources	Obfuscate Control Flow	Exception Obfuscation	Incremental Obfuscation	Obfuscate Native code	String Encryption	Custom Encrypton	Bytecode Obfuscation	Remove Debug Info	Code Optimization	Water-marking	Call Hiding (Reflection)	Commercial	Free
Dexguard	√	√	√			√	√				√		√	√	
DashO	√		√		√		√	√			√	√		√	
ProGuard	√									√	√				√
yGuard	√										√				√
KlassMaster	√		√	√	√		√				√			√	
Shield4J		√					√		√		√			√	
Jshrink	√						√				√			√	
Allatori	√		√				√			√	√	√		√	
GuardIT for Java	√	√					√		√	√			√	√	

CONTRIBUTION & FUTURE WORK

The proposed method detects obfuscation tools like ProGurad (renaming feature), Allatori (string encryption), Dasho (string encryption) and packers like bangcle, 360, baidu, tencent, ijiami. The future work will include designing and implementation of detection techniques for other obfuscation features and to publish the obfuscated dataset with different tools and options for research community.