DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master’s Thesis in Informatics

Analysis of Android Cracking Tools and Investigations in Counter Measurements for Developers

Johannes Neutze, B. Sc.

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Analysis of Android Cracking Tools and Investigations in Counter Measurements for Developers

Analyse von Android Crackingtools und Untersuchung geeigneter Gegenmaßnahmen für Entwickler

Author:	Johannes Neutze, B. Sc.
Supervisor:	Prof. Dr. Uwe Baumgarten
Advisor:	Nils Kannengießer, M. Sc.
Submission Date:	March 15, 2015

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, March 15, 2015 Johannes Neutze, B. Sc.

Acknowledgments

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Assumptions

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Abstract

`http://users.ece.cmu.edu/~koopman/essays/abstract.html` Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Contents

Glossary

.class Java Byte Code produced by the Java compiler from a .java file.

.dex Dalvik Byte Code file, translated from the Java bytecode. Dalvik Executables are designed to run on system with memory or processor constraints. For example, the .dex file of the Phone application is inside the system/app/Phone.apk.

.odex Optimized Dalvik Byte Code file are Dalvik Executables optimized for the current device the application is running on. For example, the .odex file of the Phone application is system/app/Phone.odex.

ADB The Android Debug Bridge is a command-line application providing different debugging tools.

APK An Android Application Package is the file format used for distributing and installing applications on the Android operating system. It contains the applications assets, code (.dex file), manifest and resources.

assembler Ein Assembler (auch Assemblierer[1]) ist ein Computerprogramm, das Assemblersprache in Maschinensprache übersetzt, beispielsweise den Assemblersprachentext „CLI“ in den Maschinensprachentext „11111010“..

disassembler Ein Disassembler ist ein Computerprogramm, das die binär kodierte Maschinensprache eines ausführbaren Programmes in eine für Menschen lesbarere Assemblersprache umwandelt. Seine Funktionalität ist der eines Assemblers entgegengesetzt..

Acronyms

.dex Dalvik EXecutable file.

.odex Optimized Dalvik EXecutable file.

ADB Android Debug Bridge.

APK Android Application Package.

SDK Software Development Kit.

TUM Technische Universität München.

1 Introduction

This is my real text! Rest might be copied or not be checked!

1.1 Licensing

This is my real text! Rest might be copied or not be checked!

Was ist licensing?
Ziele von Licensing
was für möglichkeiten gibt es (lvl, amazon, samsung)

1.2 Motivation

This is my real text! Rest might be copied or not be checked!

Piracy
lose money from sale/IAP
lose ad revenues
others earn the money - ad ID replacement
no control at all when cracked and in other markets -> no fixes/updates (https://youtu.be/TNnccRimhsI?t=

for user: when downloading pirated apk, no idea what they changed (malware, stealing data,privacy, permissions)
wont notice any difference since in backgound
unpredicted traffic for your server, be prepared to block pirated traffic
cracking can lead to bad user experience, e.g. copied apps, mostly for paid apps

awesome algorithms can be stolen

similar problems with inapp billing

best way to counter: license verification libraries encryption can be dumped from memory

generell piracy!!!

enthält als Abschluss SCOPE

1.3 Related Work

This is my real text! Rest might be copied or not be checked!

related work

2 Foundation

Before discussing the counter measurements, necessary basics have to be explained. This includes piracy in general as well as its relationship with Android.

2.1 Software Piracy

This is my real text! Rest might be copied or not be checked!
`http://www.xda-developers.com/piracy-testimonies-causes-and-prevention/`

2.1.1 Overview

This is my real text! Rest might be copied or not be checked!

Definition of Software Piracy?

This is my real text! Rest might be copied or not be checked!

History of Software Piracy

This is my real text! Rest might be copied or not be checked!

ewiges hin und her wer besser ist, siehe computer spiele, streaming drm, generell alles wo DRM ist

Forms of Software Piracy

This is my real text! Rest might be copied or not be checked!

Release Groups, blackmarket, app beispiele, foren etc

2.1.2 Threat to Developers

This is my real text! Rest might be copied or not be checked!

scahden für entwickler (ad id klau,)

2.1.3 Risks to Users

This is my real text! Rest might be copied or not be checked!

malware, bad user experience
It is not unlikely for a malware developer to abuse existing applications by injection of malicious functionalities and consequent redistribution of the trojanized versions C.A. CastilloandMobileSecurityWorkingGroupMcAfee,\T1\textquotedblleftAndroidmalwarepast, present,andfuture,\T1\textquotedblright2011.

2.1.4 Piracy on Android

This is my real text! Rest might be copied or not be checked!

Poor model, since self-signed certificates are allowed, 27 http://newandroidbook.com/files/Andevcon-Sec.pdf
http://www.fiercedeveloper.com/story/preventing-android-applications-piracy-possible-requi
2012-08-14 piracy umfrage on android
übergehen zu wie android funktioniert und warum es dort piracy gibt

2.2 Android

flow wie funktioniert android und warum ist es so einfach zu piraten
This is my real text! Rest might be copied or not be checked!

https://net.cs.uni-bonn.de/fileadmin/user_upload/plohmann/2012-Schulz-Code_Protection_in_Android.pdf

2.2.1 Introduction

This is my real text! Rest might be copied or not be checked!

What is Android? Where is it used? When was it founded? Who does it belong to?

platform developed by the "Android Open Source Project" -> official website?
currently one of the main operating systems for mobile devices -> quelle
focussed on simplicity for users, install of apps etc -> quelle

riesiger markt

2.2.2 Evolution of the Android Compiler

This is my real text! Rest might be copied or not be checked!

2.2.3 Build Process of Android Applications

This is my real text! Rest might be copied or not be checked!

für 4.1 erklären, bild `http://developer.android.com/tools/building/index.html`

Aufbau APK erklären

many steps and tools until the APK is build and ready to be deployed
applications are written in the Java programming language by the developer, code is available in .class files
step 1: Java files which will be compiled into .class files by a Java Compiler, similar to a Java program build process, class files contain Java bytecode representing the compiled application, optional step apply a Java Obfuscator
step 2: transformation from Java bytecode into Dalvik bytecode, see oben, dx programm from android sdk (due to it is necessity for building an application for the Android platform), output saved in singel .dex file, included in an APK in next step, possible to apply a further obfuscator operating on Dalvik bytecode(ERKLÄRUNG)
step 3: packing and signing the APK, ApkBuilder constructs an apk file out of the "classes.dex" file and adds further resources like images and ".so" files, ".so" files are shared objects which contains native functions that can be called from within the DVM, jarsigner adds developers signature to APK(ERKLÄREN WARUM SIGNATURE NÖTIG UND WO GEPRÜFT), now the app can be installed

Android applications are written in the Java [11] programming language and deployed as files with an ".apk" suffix, later called APK. It is basically a ZIP-compressed file and contains resources of the application like pictures and layouts as well as a dex file
This dex file, saved as "classes.dex", contains the program code in form of Dalvik bytecode. Further explanations on this bytecode format are given in section 3.2. The content of the APK is also cryptographically signed, which yields no security improvement but helps to distinguish and confirm authenticity of different developers of Android applications.
DIe apk kann dann per adb, market oder direkt installiert werden

Within the installation process, every installed application gets its own unique user ID (UID) by default. This means that every application will be executed as a separate system user. -> QUELLE/SINN?

2.2.4 Original Copy Protection

This is my real text! Rest might be copied or not be checked!

It replaces the old system copy protection system, wherein your APKs would be put in a folder that you can't access. Unless you root. Oh, and anyone who can copy that APK off can then give it to someone else to put on their device, too. It was so weak, it was almost non-existant.
kann mit root umgangen werden

Im original vom Markt direkt rutnergeladen und dann wird sie an den ort geschoben und kann nicht mehr zugegriffen werden -> rechte etc, QUELLE

2.2.5 Root on Android

This is my real text! Rest might be copied or not be checked!

what is it? how is it achieved? what can i do with it? (good/bad sides)
Android insecure, can be rooted and get apk file `http://androidvulnerabilities.org/all` search for root

2.3 License Verification Libraries

This is my real text! Rest might be copied or not be checked!

spreading causes giant market (zahlen) -> google play with big financial value (zahlen)
je größer der markt desto attraktiver für cracker da auch mehr leute die app ggfs gecracked runterladen würden (genauere argumente)
deswegen auch gesteigertes interesse bei developern ihre app und IP zu schützen
da wie oben gesehen die original copyprotection ausgehebelt ist, muss etwas neues her auch für den markt leiter sit es interessant

This chapter contains the LVL which will be looked at
What is a lvl? why are they used? connection to store
`http://www.digipom.com/how-the-android-license-verification-library-is-lulling-you-into-a`
dependent on store

major players have own stores and thus own lvl

Since the original approach of subsection 2.2.4 was voided, another method had to be introduced. First looks great, puts the copy protection inside the app, a from of DRM communicate with server, authorize use of application
does not prevent user from copying/transfering app, but copy useless since the app does run without the correct account
google die ersten, andere folgen, anfangs problem, dass dadurch nur durch google store geschützt war, grund dafür dass evtl ein programmierer in meinen store kommt

2.3.1 Amazon

This is my real text! Rest might be copied or not be checked!
amazon drm kiwi

Implementation

This is my real text! Rest might be copied or not be checked!
done by amazon packaging tool
release `http://www.androidheadlines.com/2010/10/amazon-send-developers-a-welcome-package.html`
`https://developer.amazon.com/public/support/submitting-your-app/tech-docs/submitting-your-app`

Functional Principle

This is my real text! Rest might be copied or not be checked!
sis is text was sind voraussetzungen? amazon app, account active der die app hat

Example

This is my real text! Rest might be copied or not be checked!
anhand eigener app

2.3.2 Google

This is my real text! Rest might be copied or not be checked!

License Verification Library
http://www.digipom.com/how-the-android-license-verification-library-is-lulling-you-into-a
related https://developers.google.com/games/services/android/antipiracy
http://android-developers.blogspot.de/2010/09/securing-android-lvl-applications.html
http://developer.android.com/google/play/licensing/overview.html
problem http://daniel-codes.blogspot.de/2010/10/true-problem-with-googles-license.html
The LVL library only works on apps sold through Google's Android Market Release
http://android-developers.blogspot.de/2010/07/licensing-service-for-android.html

Implementation

This is my real text! Rest might be copied or not be checked!

sis is text

Functional Principle

This is my real text! Rest might be copied or not be checked!

how does google license check work http://android.stackexchange.com/questions/22545/how-does-google-plays-market-license-check-work
sis is text

was sind voraussetzungen? googel acc auf dem smartphone welcher die app gekauft hat

bild http://android-developers.blogspot.de/2010/07/licensing-service-for-android.html

Example

This is my real text! Rest might be copied or not be checked!

anhand eigener app

2.3.3 Samsung

This is my real text! Rest might be copied or not be checked!

Zirconia http://developer.samsung.com/technical-doc/view.do?v=T000000062L

Implementation

This is my real text! Rest might be copied or not be checked!

sis is text `http://developer.samsung.com/technical-doc/view.do?v=T000000062L`

Functional Principle

This is my real text! Rest might be copied or not be checked!

sis is text

was sind voraussetzungen?

Example

This is my real text! Rest might be copied or not be checked!

anhand eigener app

`http://developer.samsung.com/technical-doc/view.do?v=T000000062L`

2.4 Analysis

The Cracking Tool has to alter an application's behaviour by applying patches only to the Android Application Package (APK) file, since it is the only source of code on the phone. This is the reason for the investigations to start with analysing the APK. This is done using static analysis tools. The aim is to get an accurate overview of how the circumventing of the license verification mechanism is achieved. This knowledge is later used to find counter measurements to prevent the specific Cracking Tool from succeeding.

The reengineering has to be done by using different layers of abstraction. The first reason is because it is very difficult to conclude from the altered bytecode, which is not human-readable, to the new behaviour of the application. The second reason is because the changes in the Java code are interpreted by the decompiler, which might not reflect the exact behaviour of the code or even worse, cannot be translated at all.

These problems are encountered by analysing the different abstraction levels of code as well as different decompilers.

gaining information about a program and its implementation details, process aims at enabling an analyst to understand the concrete relation between implementation and functionality, optimal output of such a process would be the original source code of the application, not possible in general

Therefore, it is necessary for such a process to provide on the one hand abstract information about structure and inter-dependencies and on the other hand result in very detailed information like bytecode and mnemonics that allow interpretation of implementation
hoffentlich starting points für investigations
java, e.g. read the program code faster

was ist reengineering? wie funktioniert es? was ist das ziel?
reverse engineering process makes use of a whole range of different analysis methodologies and tools.
only consider static analysis tools

IN ORDER TO GET FULL OVERVIEW DEX/SMALI/JAVA -> WARUM?

WAS MACHEN DIE TOOLS IM ALLGEMEINEN? WOZU BENUTZEN WIR SIE?

`https://mobilesecuritywiki.com/`
`https://net.cs.uni-bonn.de/fileadmin/user_upload/plohmann/2012-Schulz-Code_Protection_in_Android.pdf`
main tools

Getting an APK

In the following there will be an example application to generalise the procedure. The application is called L̈icenseTeständ has for our purpose a license verification library included (Amazon, Google or Samsung).
In order to analyse an APK, it has to be pulled from the Android device onto the computer. First the package name of the app has to be found out. This can be done by using the Android Debug Bridge (ADB). Entering example 1 returned example 2

adb shell ṕm list packages -f1́ example 2 enthällt return von 1

adb pull /data/app/me.neutze.licensetest-1/ /Downloads

Dex

This is my real text! Rest might be copied or not be checked!

jedes tool:
woher kommt es?
wozu wurde es erfunden?

wer hat es erfunden? quelle
blabla von der seite
wozu benutze ich es?
welches abstrahierungslevel
beispiel
additional features?
WARUM SCHAUEN WIR ES UNS AN?
wo findet man es?
welches level?
vorteil
blabla aus dem internet
für menschen unintuitiv

hexadeximal value, represent exactly one instruction
unintuitiv, instruction set well defined
ggf bezug zu DALVIK/buildprocess The Dalvik Virtual Machine (DVM) provides also the ability to call native functions within shared objects out of the Dalvik bytecode. When speaking of reverse engineering an Android application we mostly mean to reverse engineer the bytecode located in the dex file of this application.

```
a6 8e 15 00 bd 8e 15 00 d5 8e 15 00 f0 8e 15 00  |................|
```

Code Snippet 2.1: Quelle

mein custom script erklären

bytecode format http://source.android.com/devices/tech/dalvik/dalvik-bytecode.html

Smali Code

This is my real text! Rest might be copied or not be checked!

stichwort mnemonics, eine seite dex und auf der anderen seite smali, dex bytecode vs smali, Only a few pieces of information are usually not included like the addresses of instructions
unintuitive representation, deswegen smali mit corresponding mnemonics
mnemonics and vice versa is available due to the bijective mapping
correct startaddress and offset can be challenging. There are two major approaches: linear sweep disassembling and recursive traversal disassembling, The linear sweep algorithm is prone to producing wrong mnemonics e.g. when a assembler inlines data so that instructions and data are interleaved. The recursive traversal algorithm is not

prone to this but can be attacked by obfuscation techniques like junkbyte insertion as discussed in section 4.4. So for obfuscation, a valuable attack vector on disassembling is to attack the address finding step of these algorithms

https://github.com/JesusFreke/smali

Smali code is the generated by disassembling Dalvik bytecode using baksmali. The result is a human-readable, assambler-like code

The smali [7] program is an assemblerhas own disassembler called "baksmali"
can be used to unpack, modify, and repack Android applications
interesting part for obfuscation and reverse engineering is baksmali. baksmali is similar to dexdump but uses a recursive traversal approach to find instructions
vorteil? -> So in this approach the next instruction will be expected at the address where the current instruction can jump to, e.g. for the "goto" instruction. This minimizes some problems connected to the linear sweep approach. baksmali is also used by other reverse engineering tools as a basic disassembler

selbe wie dex, jedoch human readable, no big difference

wo findet man es?
wie ist es erstellt?
informationsverlust?
vorteil
blabla aus dem internet

jedes tool:
woher kommt es?
wozu wurde es erfunden?
wer hat es erfunden? quelle
blabla von der seite
wozu benutze ich es?
welches abstrahierungslevel
beispiel
additional features?
ggf bezug zu DALVIK/buildprocess

wo findet man es?
welches level?
vorteil
blabla aus dem internet

Java

This is my real text! Rest might be copied or not be checked!

probleme des disassemlbers erklären
interpretations sache
deswegen zwei compiler
unterschiedliche interpretation resultiert in flow und auch ob sies können ist unterschiedlich

ectl unterschiede/vor-nachteile
ggf bezug zu DALVIK/buildprocess (Java wird disassembled und dann assembler)

Androguard

This is my real text! Rest might be copied or not be checked!

`https://github.com/androguard/androguard`

powerful analysis tool is Androguard
includes a disassembler and other analysis methods to gather information about a program
Androguard helps an analyst to get a good overview by providing call graphs and an interactive interface -> habe nur CLI benutzt
The integrated disassembler also uses the recursive traversal approach for finding instructions like baksmali, see section 2.2
one most popular analysis toolkits for Android applications due to its big code base and offered analysis methods -> quelle, warum

jedes tool:
woher kommt es?
wozu wurde es erfunden?
wer hat es erfunden? quelle
blabla von der seite
wozu benutze ich es?
welches abstrahierungslevel
beispiel
additional features?

jadx

This is my real text! Rest might be copied or not be checked!

https://github.com/skylot/jadx
jedes tool:
woher kommt es?
wozu wurde es erfunden?
wer hat es erfunden? quelle
blabla von der seite
wozu benutze ich es?
welches abstrahierungslevel
beispiel
additional features?

Comparison of Code

This is my real text! Rest might be copied or not be checked!
vergleich gibts guten einblick was geändert wurde und wie es auf dem gegebenem lvl funtkioniert

vergleich von original und modifizierten code einer apk auf einer code ebene needed to see differences before and after cracking tool

diff is used

https://wiki.ubuntuusers.de/diff
-N: Treat absent files as empty; Allows the patch create and remove files.
-a: Treat all files as text; Allows the patch update non-text (aka: binary) files.
-u: Set the default 3 lines of unified context; This generates useful time stamps and context.
-r: Recursively compare any subdirectories found; Allows the patch to update subdirectories.
script erklären

wo findet man es?
welches level?
vorteil
blabla aus dem internet

3 Cracking Android Applications with LuckyPatcher

This is my real text! Rest might be copied or not be checked!

There are plenty of applications which can be used to modify Android apps. This thesis focuses on the on on device cracking application LuckyPatcher, especially on its license verification bypassing mechanism.

3.1 What is LuckyPatcher and what is it used for?

This is my real text! Rest might be copied or not be checked!

wer hat ihn geschrieben?
auf welcher version basiere ich
su nicht vergessen
was kann er alles
was schauen wir uns an?

LuckyPatcher is described as following on the offical webpage: "Lucky Patcher is a great Android tool to remove ads, modify apps permissions, backup and restore apps, bypass premium applications license verification, and more. To use all features, you need a rooted device." [1]

install apk from palystore -> have root -> open lucky -> chose mode

LuckyPatcher is an Android Cracking Tool which can be downloaded at `http://lucky-patcher.netbew.com/`.

similar cracking tools:
or manual: decompile and edit what ever you want

3.2 Modus Operandi

This is my real text! Rest might be copied or not be checked!

wo arbeitet er?
warum dex und nicht odex anschauen?
patterns und patching modes grob erklären (modi von luckypatcher die verschiedene operationen (pattern) auf app anwenden) => vorgehensweise zur

Since the code is modified directly a static analysis is sufficient.

UM ES EINFACHER ZU MACHEN, KEINE ODEX (WARUM), APK CREATEN UND AUF EINEM NORMALEN HANDY INSTALLIEREN(dann sieht man dass man die app wem anders gecracked geben kann - ringschluss blackmarket)

3.3 What are Patching Modi are there and what do they do?

This is my real text! Rest might be copied or not be checked!

kombination von patterns.
welche modes gibt es? welche patterns benutzen sie?
welche apps getestet und welche results?

3.4 What patterns are there and what do they do?

This is my real text! Rest might be copied or not be checked!

was greift jedes pattern an? wie wird der mechanismus ausgeklingt? was ist das result?

3.5 Learnings from LuckyPatcher

This is my real text! Rest might be copied or not be checked!

was fällt damit weg?
erklären warum (2) 5.1.2 Opaque predicates zb nicht geht, da auf dex ebene einfach austauschbar
simple obfuscation for strings? x -> string (damit name egal)

4 Counter Measurements for Developers

Now that that the functionality of LuckyPatcher is analyzed, it is time to investigate in possible solutions for developers. Counter measurements preventing the cracking app from circumventing the license check mechanism are addressed in four different ways. The first chapter covers functions to discover preconditions in the environment cracking apps use to discover weaknesses or need to be functional. The second chapter uses the aquired knowledge about LuckyPatcher to modify the code resulting in the patching being unsuccessful. In the third chapter presents methods to prevent the reengineering of the developerś application and thus the creation of custom cracks. Further hardware and external measurements are explained in the fourth chapter.
general suggestions by google `http://android-developers.blogspot.de/2010/09/securing-android-lvl-applications.html`

4.1 Tampering Protection

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN DASS IM PROGRAMMIER PROZESS IMPLEMENTIERT

Environment and Integrity Checks, wenn die umgebung falsch ist, kann die app verändert werden. deswegen von vornherein ausschließen, dass die bedingungen dafür gegeben sind.
siehe masterarbeit 2

mechanisms should work for amazon/lvl/samsung –> beweis! (amazon die signature den die seite vorgibt?)

force close im falle von falschem outcome, entspricht nicht android qualität `http://developer.android.com/distribute/essentials/quality/core.html` aber so wird es dem user klarer dass seine application gecracked ist. harmlosere variante dialog anzeigen oder element nicht laden.

es gibt verschieden punkte um die integrity der application sicherzustellen. dies beinhaltet die umgebung debugg oder rootzugriff, die suche nach feindliche installierte applicationen oder checks nach der rechtmäßigen installation und rechtmäßigen code.

4.1.1 Prevent Debuggability

This is my real text! Rest might be copied or not be checked!

WAS IST DIE IDEE DAHINTER? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

der debug modus kann dem angreifer informationen/logs über die application geben während diese läuft, aus diesen informationen können erkenntnisse über die funktionsweise geben die für einen angriff/modifikation gewonnen werden können. aus diesen informationen können dann patches für software wie lucky patcher entwickelt werden, da man die anzugreifenden stellen bereits kennt. kann erzwungen werden indem man das debug flag setzt (wo ist es, wie kann es gesetzt werden)
um dies zu verhindern kann gecheckt werden ob dieses flag forciert wird und gegebenenfalls das laufen der application unterbinden

```
14  public static boolean isDebuggable(Context context) {
15      boolean debuggable = (0 != (context.getApplicationInfo().flags & ApplicationInfo.
            FLAG_DEBUGGABLE));
16
17      if (debuggable) {
18          android.os.Process.killProcess(android.os.Process.myPid());
19      }
20
21      return debuggable;
22  }
```

Code Snippet 4.1: asd[1]

Code SNippet /refCode Snippet: luckycode zeigt eine funktion die auf den debug modus prüft. Dazu werden zuerst in zeile 15 die appinfo auf das debug flag überprüft. ist dieses vorhanden, ist die variable debuggable true. in diesem fall wird dann die geschlossen

4.1.2 Root Detection

This is my real text! Rest might be copied or not be checked!

WAS IST DIE IDEE DAHINTER? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

http://stackoverflow.com/questions/10585961/way-to-protect-from-lucky-patcher-play-licen

```
    public static boolean findBinary(Context context, final String binaryName) {
        boolean result = false;
        String[] places = {
                "/sbin/",
                "/system/bin/",
                "/system/xbin/",
                "/data/local/xbin/",
                "/data/local/bin/",
                "/system/sd/xbin/",
                "/system/bin/failsafe/",
                "/data/local/"
        };

        for (final String where : places) {
            if (new File(where + binaryName).exists()) {
                result = true;
                android.os.Process.killProcess(android.os.Process.myPid());
            }
        }

        return result;
    }
```

Code Snippet 4.2: Partial Listing

SafetyNet provides services for analyzing the configuration of a particular device, to make sure that apps function properly on a particular device and that users have a great experience. https://developer.android.com/training/safetynet/index.html
Checking device compatibility with safetynet
Unlocked bootloader doesn't matter. Can't have root installed initially. Has to be a stock / signed ROM. https://www.reddit.com/r/Android/comments/3kly2z/checking_device_compatibility_with_safetynet/

4.1.3 LuckyPatcher Detection

This is my real text! Rest might be copied or not be checked!
WAS IST DIE IDEE DAHINTER? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

As the example shows, this check is not only a solution to prevent the application from running when LuckyPatcher is present on the device. The screening can be expanded to check for the installation of any other application, like black market apps or other cracking tools as the code example Code Example 4.5 shows.

`http://stackoverflow.com/questions/13445598/lucky-patcher-how-can-i-protect-from-it` `http://android-onex.blogspot.de/2015/07/anti-piracy-software-activated-solved.html`

```java
    public static boolean checkInstall(final Context context) {
        boolean result = false;
        String[] luckypatcher = new String[]{
                // Lucky patcher
                "com.dimonvideo.luckypatcher",
                // Another lucky patcher
                "com.chelpus.lackypatch",
                // Black Mart alpha
                "com.blackmartalpha",
                // Black Mart
                "org.blackmart.market",
                // Lucky patcher 5.6.8
                "com.android.vending.billing.InAppBillingService.LUCK",
                // Freedom
                "cc.madkite.freedom",
                // All−in−one Downloader
                "com.allinone.free",
                // Get Apk Market
                "com.repodroid.app",
                // CreeHack
                "org.creeplays.hack",
                // Game Hacker
                "com.baseappfull.fwd"
        };

        for (String string : luckypatcher) {
            if(checkInstallerName(context, string)){
                result = true;
            }

            if (result) {
                android.os.Process.killProcess(android.os.Process.myPid());
            }
        }

        return result;
    }

    private static boolean checkInstallerName(Context context, String string) {
        PackageInfo info;
        boolean result = false;

        try {
            info = context.getPackageManager().getPackageInfo(string, 0);

            if (info != null) {
                android.os.Process.killProcess(android.os.Process.myPid());
                result = true;
            }

        } catch (final PackageManager.NameNotFoundException ignored) {
        }

        if (result) {
            android.os.Process.killProcess(android.os.Process.myPid());
        }
        return result;
    }
}
```

Code Snippet 4.3: Partial Listing

4.1.4 Sideload Detection

This is my real text! Rest might be copied or not be checked!

WAS IST DIE IDEE DAHINTER? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

http://stackoverflow.com/questions/10809438/how-to-know-an-application-is-installed-fro

```
public class Sideload {
    private static final String PLAYSTORE_ID = "com.android.vending";
    private static final String AMAZON_ID = "com.amazon.venezia";
    private static final String SAMSUNG_ID = "com.sec.android.app.samsungapps";

    public static boolean verifyInstaller(final Context context) {
        boolean result = false;
        final String installer = context.getPackageManager().getInstallerPackageName(context.
             getPackageName());

        if (installer != null) {
            if (installer.startsWith(PLAYSTORE_ID)) {
                result = true;
            }
            if (installer.startsWith(AMAZON_ID)) {
                result = true;
            }
            if (installer.startsWith(SAMSUNG_ID)) {
                result = true;
            }
        }
        if(!result){
            android.os.Process.killProcess(android.os.Process.myPid());
        }

        return result;
    }
```

Code Snippet 4.4: Partial Listing

4.1.5 Signature

This is my real text! Rest might be copied or not be checked!

http://developer.android.com/tools/publishing/app-signing.html
http://forum.xda-developers.com/showthread.php?t=2279813&page=5

CONTRA

Local Signature Check

This is my real text! Rest might be copied or not be checked!

WAS IST DIE IDEE DAHINTER? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

local check whether signature is allowed
once in code
save to use signature in code?

```
    public static boolean checkAppSignature(final Context context) {
        //Signature used to sign the application
        static final String mySignature = "...";
        boolean result = false;

        try {
            final PackageInfo packageInfo = context.getPackageManager().getPackageInfo(context.
                getPackageName(), PackageManager.GET_SIGNATURES);

            for (final Signature signature : packageInfo.signatures) {
                final String currentSignature = signature.toCharsString();
                if (mySignature.equals(currentSignature)) {
                    result = true;
                }
            }
        } catch (final Exception e) {
            android.os.Process.killProcess(android.os.Process.myPid());
        }

        if (!result) {
            android.os.Process.killProcess(android.os.Process.myPid());
        }

        return result;
    }
```

Code Snippet 4.5: Partial Listing

Remote Signature Verification and Remote Code Loading

This is my real text! Rest might be copied or not be checked!

WAS IST DIE IDEE DAHINTER? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

certificate an server, get signature and send to server
content direkt von server laden (e.g. all descriptions, not sure if dex possible)
maps checks for signature?
e.g. account auf seite erstellen, ecrypted dex ziehen der von loader stub geladen wird (like packer) kann wiedermal dann gezogen werden und dann als custom patch verteilt werden

4.2 LVL Modifications

This is my real text! Rest might be copied or not be checked!

ERWÄHNEN DASS IM PROGRAMMIER PROZESS IMPLEMENTIERT

siehe masterarbeit 2 `http://www.digipom.com/how-the-android-license-verification-library-is-`
What can I do?
Googleś License Verification Library is modified in a way

4.2.1 Modify the Library

This is my real text! Rest might be copied or not be checked!

google

WAS IST DIE IDEE DAHINTER? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

4.2.2 Checken ob ganzer code abläuft und dann nacheinander elemente aktivieren

This is my real text! Rest might be copied or not be checked!

WAS IST DIE IDEE DAHINTER? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

master1 - testen

damit die ganzen blöcke durchlaufen werden müssen

4.2.3 Native Implementierung

This is my real text! Rest might be copied or not be checked!

Library native
native ist nicht in dex -> besser?
so nur schnittstelle? dann einfach sozusagen die schnittstelle faken

luckypatcher tauscht so ganz aus bzw modifizeirt dex
ggf einfach .so library austauschen, aber großer aufwand zu kompilieren davor

4.3 Preventing Reengineering

This is my real text! Rest might be copied or not be checked!

Application developers are interested in protecting their applications. Protection in this case means that it should be hard to understand what an application is doing and how its functionalities are implemented.

Reverse engineering of Android applications is much easier than on other architectures -> high level but simple bytecode language

Obfuscation techniques protect intellectual property of software/license verification

possible code obfuscation methods on the Android platform focus on obfuscating Dalvik bytecode -> limitations of current reverse engineering tools

https://blog.fortinet.com/post/how-android-malware-hides
http://www.hotforsecurity.com/blog/mobile-app-development-company-fights-off-android-malwa
html

4.3.1 Basic Breaks for Common Tools

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN WO IM PROZESS ANGEWENDET

pros and cons sagen?
https://github.com/strazzere/APKfuscator
http://www.strazzere.com/papers/DexEducation-PracticingSafeDex.pdf
https://youtu.be/Rv8DfXNYnOI?t=811

Filesystem

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN WO IM PROZESS ANGEWENDET

make classname to long
https://youtu.be/Rv8DfXNYnOI?t=985 works except for the class
breaks only baksmali

Inject bad OPcode or Junkbytes

This is my real text! Rest might be copied or not be checked!
viele schlagen vor: junkbyte injection well known technique in x86, confuse disassemblers in a way that they produce disassembly errors and disallow correct interpretations, inserting junkbytes in selected locations within the bytecode where a disassembler expects an instruction, junkbyte must take the disassembling strategy into account in order to reach a maximum of obfuscated code, break the two disassebly stragien from

2.4.0, condition for the location is that the junkbyte must not be executed, because an execution would result in an undesired behavior of the application, junkbyte must be located in a basic block which will never be executed [3]

funktioniert nicht mehr This is not expected to work. The bytecode verifier explicitly checks all branches for validity. The question of whether or not an address is an instruction or data is determined by a linear walk through the method. Data chunks are essentially very large instructions, so they get stepped over.

You can make this work if you modify the .odex output, and set the "pre-verified" flag on the class so the verifier doesn't examine it again – but you can't distribute an APK that way.

This "obfuscation" technique worked due to an issue in dalvik. This issue was fixed somewhere around the 4.3 timeframe, although I'm not sure the first released version that contained the fix. And lollipop uses ART, which never had this issue.

Here is the change that fixed this issue: `https://android-review.googlesource.com/#/c/57985/` `http://stackoverflow.com/questions/33110538/junk-byte-injection-in-android`
ERWÄHNEN WO IM PROZESS ANGEWENDET

JUNKBYTES
use bad opcode in deadcode
code runs but breaks tools
put it into a class you do not use –> care proguard, it will not use it since it is not included
–> fixed...
`https://youtu.be/Rv8DfXNYnOI?t=1163`
reference not inited strings
`https://youtu.be/Rv8DfXNYnOI?t=1459`

Throw exceptions which are different in dalvik than in java

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN WO IM PROZESS ANGEWENDET

recursive try/catch? –> valid dalvik code
`https://youtu.be/Rv8DfXNYnOI?t=1650`

Increase headersize

This is my real text! Rest might be copied or not be checked!

ERWÄHNEN WO IM PROZESS ANGEWENDET

you have to edit every other offset as well
`https://youtu.be/Rv8DfXNYnOI?t=1890`
dexception, dex within a dex by shifting
this is a packer/encrypter
slowdown automatic tools
`https://youtu.be/Rv8DfXNYnOI?t=1950`

Endian Tag

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN WO IM PROZESS ANGEWENDET

reverse endian
breaks tools works on device (odex)
lot work for little gain
`https://youtu.be/Rv8DfXNYnOI?t=2149`

4.3.2 Optimizors and Obfuscators

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN WO IM PROZESS ANGEWENDET

Obfuscators/Optimizors definition
Obfuscation techniques are used to protect software and the implemented algorithms designed to make reverse engineering harder and more time consuming, hin und her zwischen obfuscation und reverse engineering techniken
obfuscation techniques must not alter the behavior of programs, often only target specific reverse engineering steps, few general protection schemes, possible slower execution, not topic here, just examples for obfuscation applications
remove dead/debug code
potentially encrypt/obfuscate/hide via reflection
`https://youtu.be/6vFcEJ2jgOw?t=243`

definition obfuscation, was macht es, wie funktioniert es, wer hat es erfunden, wie wendet man es an "hard to reverse engineer" but without changing the behavior of this application, was heißt hardto reverse

parallele zu disassembler ziehen

Obfuscation cannot prevent reverse engineering but can make it harder and more time consuming. We will discuss which obfuscation and code protection methods are applicable under Android and show limitations of current reverse engineering tools

The following optimizers/obfuscators are common tools. (dadrin dann verbreitung preis etc erklären)

Proguard

This is my real text! Rest might be copied or not be checked!
`https://youtu.be/6vFcEJ2jgOw?t=419`

WER HAT ES HERGESTELLT? WAS IST ES? WAS SIND DIE FEATURES? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

identifier mangling, ProGuard uses a similar approach. It uses minimal lexical-sorted strings like a, b, c, ..., aa, ab, original identifiers give information about interesting parts of a program, Reverse engineering methods can use these information to reduce the amount of program code that has to be manually analyzed -> neutralizing these information in order to prevent this reduction, remove any meta information about the behavior, meaningless string representation holdin respect to consistence means identifiers for the same object must be replaced by the same string, advantage of minimizing the memory usage, e development process in step ”a” or step ”b”
string obfuscationa, string must be available at runtime because a user cannot understand an obfuscated or encrypted message dialog, information is context, other is information itself, e.g. key, url, injective function and deobfuscation stub which constructs original at runtime so no behaviour is changed, does not make understanding harder since only stub is added but reduces usable meta information **[SchulzCodeProtection]**
EVALUATION: dynamic analysis beats this and read directly from memory

ProGuard is an open source tool which is also integrated in the Android SDK `http://proguard.sourceforge.net/` `http://developer.android.com/sdk/index.html` was ist proguard? was macht er? -> ProGuard is basically a Java obfuscator but can also be used for Android applications because they are usually written in Java // feature set includes identifier obfuscation for packages, classes, methods, and fields

was kann er noch? -> Besides these protection mechanisms it can also identify and highlight dead code so it can be removed in a second, manual step. Unused classes can be removed automatically by ProGuard.

easy integration -> how

`http://developer.android.com/tools/help/proguard.html`

optimizes, shrinks, (barely) obfuscates –> free, reduces size, faster

gutes bild `https://youtu.be/TNnccRimhsI?t=1360`

removes unnecessary/unused code

merges identical code blocks

performs optimiztations

removes debug information

renames objects

restructures code

removes linenumbers –> stacktrace annoying

`https://youtu.be/6vFcEJ2jgOw?t=470`

–>hacker factor 0

does not really help

googles commentar `http://android-developers.blogspot.de/2010/09/proguard-android-and-licens` `html`

eine art result bzw zusammenfassung -> Without proper naming of classes and methods it is much harder to reverse engineer an application, because in most cases the identifier enables an analyst to directly guess the purpose of the particular part. The program code itself will not be changed heavily, so the obfuscation by this tool is very limited.

Dexguard

This is my real text! Rest might be copied or not be checked!

ERWÄHNEN WO IM PROZESS ANGEWENDET

WER HAT ES HERGESTELLT? WAS IST ES? WAS SIND DIE FEATURES? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

master2
OVERVIEW
son of proguard
the s̈tandardp̈rotection
optimizer
shrinekr
obfuscator/encrypter, does not stop reverse engineering
`https://youtu.be/6vFcEJ2jgOw?t=643`
WHAT DOES IT DO
everything that proguard does
automatic reflection
strign encryption
asset/library encryption
class encryption(packign)
applciation tamper protection
file->automatic reflection->string encryption->file
`https://youtu.be/6vFcEJ2jgOw?t=745`
class encryption= packer, unpackers do it most of the time in few seconds, aber aufwand auf handy, nicht so einfach wie pattern in luckypatcher
CONS
may increase dex size, memory size; decrease speed
removes debug information
string, etc encryption
best feature: automatic reflection with string encryption
reversible with moderate effort
hacker protection factor 1

ESULT -> UNTERSCHIED ZU DEN VORHERIGEN -> The obfuscation methods used in Allatori(dexguard) are a superset of ProGuards so it is more powerful but does not prevent an analyst from disassembling an Android application.

Allatori

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN WO IM PROZESS ANGEWENDET

WER HAT ES HERGESTELLT? WAS IST ES? WAS SIND DIE FEATURES? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD) http://www.allatori.com/clients/index.php

Allatori is a commercial product from Smardec.
Besides the same obfuscation techniques like ProGuard, shown in section 2.1, Allatori also provides methods to modify the program code. Loop constructions are dissected in a way that reverse engineering tools cannot recognize them. This is an approach to make algorithms less readable and add length to otherwise compact code fragments. Additionally, strings are obfuscated and decoded at runtime. This includes messages and names that are normally human readable and would give good suggestions to analysts.

cannot recognize them. WHAT DOES IT
name obfuscation
control flow flattening/obfuscation
debug info obfuscation
string encryption
RESULT
decreases dex size, memory, increases speed
remvoes debug code
not much obfuscation
Proguard+string encryption
easily reversed
hacker protection factor 0.5
https://youtu.be/6vFcEJ2jg0w

https://net.cs.uni-bonn.de/fileadmin/user_upload/plohmann/2012-Schulz-Code_Protection_in_Android.pdf
Allatori [6] is a commercial product from Smardec. Besides the same obfuscation techniques like ProGuard, shown in section 2.1, Allatori also provides methods to modify the program code. Loop constructions are dissected in a way that reverse engineering tools cannot recognize them. This is an approach to make algorithms less readable and add length to otherwise compact code fragments. Additionally, strings are obfuscated and decoded at runtime. This includes messages and names that are normally human readable and would give good suggestions to analysts. The obfuscation methods used in Allatori are a superset of ProGuards so it is more powerful but does not prevent an analyst from disassembling an Android application
Allatori. Allatori obfuscator. Visited: May, 2012. [Online]. Available: http://www.allatori.com/doc.html

RESULT -> UNTERSCHIED ZU DEN VORHERIGEN -> The obfuscation methods used in Allatori are a superset of ProGuards so it is more powerful but does not prevent an analyst from disassembling an Android application.

4.3.3 Protectors and Packers

This is my real text! Rest might be copied or not be checked!

from malware

APKprotect

This is my real text! Rest might be copied or not be checked!

WER HAT ES HERGESTELLT? WAS IST ES? WAS SIND DIE FEATURES? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

stub fixes broken code which is normally not translated by tools, breaks static analysis
`https://youtu.be/6vFcEJ2jgOw?t=347`
`https://youtu.be/6vFcEJ2jgOw`
chinese protector
also known as dexcrypt, appears active but site down, clones might be available
anti-debug, anti-decompile, almost like a packer
string encryption
cost ???
tool mangles code original code
-modifies entrypoint to loader stub
-prevents static analysis
during runtime loader stub is executed
-performs anti-emulation
-performs anti-debugging
-fixes broken code in memory
FUNCTION
dalvik optimizes the dex file into momory ignoring bad parts
upon execution dalvik code initiates, calls the native code
native code fixes odex code in memory
execution continues as normal
RESULT
slight file size increase
prevents easily static analysis
hard once, easy afterwards

easily automated to unprotect
still has string encryption (like DexGuard, Allatori) afterwards
not much iteration in the last time, do not knwo if still alive
hacker protection factor 3, no public documentation, but every app is the same

Packers

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN WO IM PROZESS ANGEWENDET

dynamic code loading best would be an application is transformed by obfuscator that it does not contain any meta information or directly interpretable bytecode, not possible because DVM would not be able to read -> packing, often used in malware [2] packer transforms code that a reverse engineer cannot directly extract information, e.g. by encrypting programm data no information can be extracted without decrypting, would be bad for programm as well that is why packer uses loader stub to decrypt in runtime, soultion decrypt by hand or dynamic analysis
in general two components are created, loader stub and encrypted app, on android the encrypted dex file, easier to create as the loader stub
BILD WIE ER FUNKTIONIERT step 1 load encrypted app into memory (download from server, extract from data structure, plain file available) step 2 app file is decrypted -> original dex, can be any encryption from simple to hard, speed may slow down step 3 load decrypted dex into DVM from a bytearray, see [3] step 4 execute

[3] stellt hier basic version vor, bessere versionen ist das im folgenden
result: protection makes it hard to analyze the target application, because its bytecode is only available encrypted, decrypted version the unpacking stub has to be analyzed, great slow down, other obfuscations can be applied on stub
EVALUTAION
soultion decrypt by hand or dynamic analysis
generell: angriff nur nach 2 wenn decrypted, einfach aus memory lesen da es ja iwo hingelegt werden muss um es widerum zu laden, man muss herausfinden wo stub es herläd

ggf auch nur die eine library so laden, aber dazu muss man sie so schreiben, aber auch der cracker muss aufwand betreiben

break static analysis tools, you ahve to do runtime analysis

like UPX, stub application unpacks, decrypts, loads into memory which is normally hidden from static analysis
http://www.fortiguard.com/uploads/general/Area41Public.pdf
https://books.google.de/books?id=ACjUCgAAQBAJ&pg=PA372&lpg=PA372&dq=ijiami+integrity&source=bl&ots=NTf7YaqJiZ&sig=M5GKDCcQB5dcwXR3hjtIv8pMlAA&hl=de&sa=X&ved=0ahUKEwjH3umt1b3JAhXGLA8KHYhwDGsQ6AEIMDAC#v=onepage&q=ijiami%20integrity&f=false
https://www.blackhat.com/docs/asia-15/materials/asia-15-Park-We-Can-Still-Crack-You-Gener
pdf
https://www.virusbtn.com/conference/vb2014/abstracts/Yu.xml
https://www.virusbtn.com/pdf/conference_slides/2014/Yu-VB2014.pdf
https://www.youtube.com/watch?v=6vFcEJ2jg0w
https://books.google.de/books?id=ACjUCgAAQBAJ&pg=PA372&lpg=PA372&dq=ijiami+integrity&source=bl&ots=NTf7YaqJiZ&sig=M5GKDCcQB5dcwXR3hjtIv8pMlAA&hl=de&sa=X&ved=0ahUKEwjH3umt1b3JAhXGLA8KHYhwDGsQ6AEIMDAC#v=onepage&q=ijiami%20integrity&f=false

concept erklären und dann die beispiele nennen, nicht mehr aktiv/gecracked aber prinzip ist gut

examples for packers are

hosedex2jar
This is my real text! Rest might be copied or not be checked!
WER HAT ES HERGESTELLT? WAS IST ES? WAS SIND DIE FEATURES? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

https://youtu.be/6vFcEJ2jg0w?t=1776
PoC packer
https://github.com/strazzere/dehoser/
not available for real use
appears defunct
near zero ITW samples
mimics dexception attack from dex education 101
FUNCTION
encrypts and injects dexfile into dex header (deception)
very easy to spot
very easy to decrypt, just use dex2jar

static analysis does not work since it sees the encrypted file
on execution loader stub decrypts in memory and dumps to file system
loader stub acts as proxy and passes events to the dex file on system using a dexClassLoader
RESULT
simple PoC
slight file size increase
attempts to prevent static analysis - kind of works
lots of crashing
easily automated to unpack
easy to reverse, good for learning
hacker protection factor 0.5

Pangxie

This is my real text! Rest might be copied or not be checked!

WER HAT ES HERGESTELLT? WAS IST ES? WAS SIND DIE FEATURES? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

`https://youtu.be/6vFcEJ2jg0w?t=1982`
anti-debug
anti-tamper
appears to be defunct product
little usage/samples ITW
FUNCTION
`https://youtu.be/6vFcEJ2jg0w?t=2040`
encrypts dex file and bundles as asset in APK
very easy to find, logcat has to much information
dalvik calls JNI layer to verify and decrypt
easy to reverse, both dalvik and native, excellent for beginners to Android and packers
aes used only for digest verification
easily automated, 0x54 always the key
or dynamically grab app_dex folder
slightly increase file size
prevents static analysis - though easy to identify
uses static 1 byte key for encryption
easily automated to unpack
very easy to reverse, good for learning
good example of an unobfuscated packer stub for cloning

hacker protection faktor 1.5
only working till <4.4
simple packer, increase encryption with key, do not just dump on filesystem

BANGCLE

This is my real text! Rest might be copied or not be checked!

WER HAT ES HERGESTELLT? WAS IST ES? WAS SIND DIE FEATURES? WIE FUNKTIONIERT ES? WIE WIRD ES IMPLEMENTIERT? WIE SIEHT DAS RESULT AUS (EXAMPLE BILD)

anti-debugging
anti-tamper
anti-decompilation
anti-runtime injection
online only service, apk checked for malware
detected by some anti virus due to malware
cost 10k
no one has done it before...
stopped working on 4.4
FUNCTION
dalvik execution talks launched JNI
JNI launches secondary process
chatter over PTRACE between the two processes
newest process decrypts dex into memory
original dalvik code proxies everything to the decrypted dex
RESULT
well written, lots of anti-* tricks
seems to be well supported and active on development
does a decent job on online screening - no tool released for download (though things clearly to slip through)
not impossible to reverse and re-bundle packages
current weakness (for easy runtime unpacking) is having a predictable unpacked memory location
hacker protect faktor 5
probably best tool out there but lag when updating since online approval

4.4 External Improvements

This is my real text! Rest might be copied or not be checked!

4.4.1 Service-managed Accounts

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN WO IM PROZESS ANGEWENDET

https://youtu.be/TNnccRimhsI?t=1636
check on server what content should be returned or logic on server

kann man einen lagorithus haben um rauszufinden was man auslagern kann?

if not possible remote code loading

https://www.youtube.com/watch?v=rSH6dnUTDZo was ist dann geschützt? content, servers, time constrained urls, obfuscation by using reflection combined with SE -> makes slow but no static analysis

very very slow, e.g 10kHz so no big calculations possible
250bytes, 200ms

http://amies-2014.international-symposium.org/proceedings_2014/Kannengiesser_Baumgarten_Song_AmiEs_2014_Paper.pdf

4.4.2 ART endlich durchsetzen

This is my real text! Rest might be copied or not be checked!
ERWÄHNEN WO IM PROZESS ANGEWENDET

art hat masschinen coed
wenn reengineerbar dann nicht gut
warum jetzt noch keine art apps? https://en.wikipedia.org/wiki/Android_Runtime
dex2oat

4.4.3 Secure Elements

This is my real text! Rest might be copied or not be checked!

ERWÄHNEN WO IM PROZESS ANGEWENDET

new section trusted execution environment trusttronic letzte conference samsung knox –>gelten eher sicher

5 Evaluation of Counter Measurements

This is my real text! Rest might be copied or not be checked!

Evaluation der vorgeschlagenen punkte mit pro cons und umsetzbarkeit

http://forum.xda-developers.com/showthread.php?t=2279813

This is my real text! Rest might be copied or not be checked!

5.1 Tampering Protection

just as easy to crack as LVL when you know the code
evtl create native versions because harder to crack

5.1.1 Prevent Debuggability

5.1.2 Root Detection

5.1.3 LuckyPatcher Detection

already in some roms

https://www.reddit.com/r/Piracy/comments/3gmxun/new_way_to_disable_the_antipiracy_setting_on/
https://github.com/AlmightyMegadeth00/AntiPiracySupport
https://www.reddit.com/r/Piracy/comments/3eo8sj/antipiracy_measures_on_android_custom_roms/
example roms http://www.htcmania.com/archive/index.php/t-1049040.html

https://forums.oneplus.net/threads/patch-disable-the-antipiracy-feature-on-all-roms-sur
334772/

5.1.4 Sideload Detection

5.1.5 Signature Check

maps checks for signature?
`http://stackoverflow.com/questions/13582869/does-lucky-patcher-resign-the-app-it-patches-`
`https://developers.google.com/android/guides/http-auth` `http://forum.xda-developers.com/showthread.php?t=2279813&page=5`
generell
self signing, kann genauso geskippt werden dann remote
wenn einmal geladen -> skippen und geladene datei als custom patch nachschieben

5.1.6 Remote Verification and Code nachladen

trotzdem doof wenn einmal geladen kann man das file extrahieren etc

5.2 LVL Modifications

This is my real text! Rest might be copied or not be checked!

reengineering kann aushebeln

5.3 Prevent Reengineering

This is my real text! Rest might be copied or not be checked!

5.3.1 Packers

already cracked `https://www.google.de/search?q=hosedex2jar&oq=hosedex2jar&aqs=chrome..69i57j69i60j69i59j69i60l3.1680j0j7&sourceid=chrome&es_sm=91&ie=UTF-8`
`http://www.hotforsecurity.com/blog/mobile-app-development-company-fights-off-android-malw`
`html`

BEISPIELBILDER!! This is my real text! Rest might be copied or not be checked!

5.4 External Improvements

sis is text

5.4.1 Service-managed Accounts

5.4.2 ART

art hat oat files aber die haben dex files

5.4.3 Secure Elements

new section trusted execution environment trusttronic letzte conference samsung knox –>gelten eher sicher

6 Conclusion

This is my real text! Rest might be copied or not be checked!

research and also a valuable market for companies
Because source code can be easier recovered from an application in comparison to x86, there is a strong need for code protection and adoption of existing reverse engineering methods. Main parts of Android application functionalities are realized in Dalvik bytecode. So Dalvik bytecode is of main interest for this topic
Also, the Android system does not prevent modification of this bytecode during runtime, This ability of modifying the code can be used to construct powerful code protection schemata and so make it hard to analyze a given application.
[3]

auch wichtig weil wenn crackable dann upload zu stores und dann malware
http://www.hotforsecurity.com/blog/mobile-app-development-company-fights-off-android-malware-with-obfuscation-tool-3717.html

6.1 Summary

This is my real text! Rest might be copied or not be checked!

sis is text alles hilft gegen lucky patcher auf den ersten blick, jedoch custom patches können es einfach umgehen -> deswegen hilft nur reengineering schwerer zu machen
every new layer is another complexity

6.2 Discussion

This is my real text! Rest might be copied or not be checked!

sis is text `http://www.digipom.com/how-the-android-license-verification-library-is-lulling-y`
What Google should have really done
`http://programmers.stackexchange.com/questions/267981/should-i-spend-time-preventing-pirac`
You are asking the wrong question. Technical safeguards such as proguard are a must but are trying to solve the problem the hard way.

content driven `http://stackoverflow.com/questions/10585961/way-to-protect-from-lucky-patcher`
google sagt `http://android-developers.blogspot.de/2010/09/securing-android-lvl-applications.html`

6.3 Future Work

This is my real text! Rest might be copied or not be checked!

art?
smart cards
google vault
all papers with malware and copyright protection is interesting since they also want to hide their code

List of Figures

List of Tables

List of Code Snippets

Bibliography

[1] ChelphuS. *crapler*. URL: http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm (visited on 01/09/2016).

[2] F. Guo, P. Ferrie, and T.-c. Chiueh. "A Study of the Packer Problem and Its Solutions." English. In: *Recent Advances in Intrusion Detection*. Ed. by R. Lippmann, E. Kirda, and A. Trachtenberg. Vol. 5230. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 98–115. ISBN: 978-3-540-87402-7. DOI: 10.1007/978-3-540-87403-4_6.

[3] P. Schulz. *Code Protection in Android*. Tech. rep. Friedrich-Wilhelms-Universität Bonn, Institute of Computer Science, July 2012.