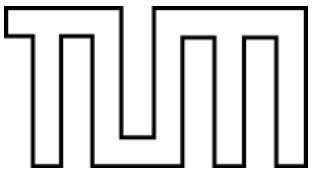


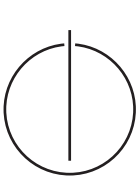
DEPARTMENT OF INFORMATICS

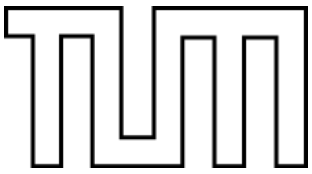
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

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

Johannes Neutze, B. Sc.





DEPARTMENT OF INFORMATICS

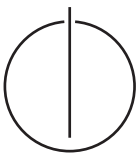
TECHNISCHE UNIVERSITÄT MÜNCHEN

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

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

API An Application Programming Interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API .

computer is a machine that. . . .

valid device a device which is allowed to run software specified by the license.

Acronyms

API Application Programming Interface.

TUM Technische Universität München.

1 Introduction

1.1 Licensing

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

1.2 Motivation

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

related work

2 Foundation

sis is a text sis is text

2.1 Software Piracy

asdasdas http://www.xda-developers.com/piracy-testimonies-causes-and-prevention/

2.1.1 Overview

Definition of Software Piracy?

History of Software Piracy

Forms of Software Piracy

Release Groups, blackmarket, app beispiele, foren etc

2.1.2 Threat to Developers

scahden für entwickler (ad id klau,)

2.1.3 Risks to Users

malware, bad user experience

2.1.4 Piracy on Android

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

2.2.1 Introduction

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

2.2.2 Evolution of the Android Compiler

sis is text

2.2.3 Original Copy Protection

Java Virtual Machine

sis is text

Dalvik Virtual Machine

sis is text

Android Runtime

im Moment abwärtskompatibilität dex in oat (tools zum extrahieren nennen)

2.2.4 Root on Android

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 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

2.3.1 Amazon

amazon drm kiwi

Implementation

sis is text

Functional Principle

sis is text

Example

anhand eigener app

2.3.2 Amazon

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

Implementation

sis is text

Functional Principle

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

Example

anhand eigener app

2.3.3 Amazon

Zirconium

Implementation

sis is text

Functional Principle

sis is text

Example

anhand eigener app

2.4 Forensicś Basics

reengineering mit forensik vergleichen und dass man im grunde das selbe wie so eine app machen kann dann

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

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

used to detect how the crackign tool modifies content in order to copy protext umgehen

can also be used to learn the code –> was nacher das allgemeine problem ist

2.4.1 Used Code Types

verschiedene level von abstrahierung
dex-smali-java
wo findet man es?
welches level?
vorteil
blabla aus dem internet

2.4.2 Forensic Tools

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

3 Cracking Android Applications with LuckyPatcher

http://lucky-patcher.netbew.com/

3.1 What is LuckyPatcher and what is it used for?

wer hat ihn geschrieben?
auf welcher version basiere ich
su nicht vergessen
was kann er alles
was schauen wir uns an?
install apk from palystore -> have root -> open lucky -> chose mode
similar cracking tools:
or manual: decompile and edit what ever you want

3.2 Operation

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

3.3 What patterns are there and what do they do?

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

3.4 What are Patching Modes are there and what do they do?

kombination von patterns.
welche modes gibt es? welche patterns benutzen sie?

welche apps getestet und welche results?

3.5 Learnings from LuckyPatcher

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

am besten mit example 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.

4.1 Tampering Protection

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

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  }
```

Listing 4.1: asd

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

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

```
16  public static boolean findBinary(Context context, final String binaryName) {
17      boolean result = false;
18      String[] places = {
19              "/sbin/",
20              "/system/bin/",
21              "/system/xbin/",
22              "/data/local/xbin/",
23              "/data/local/bin/",
24              "/system/sd/xbin/",
```

```
25                  "/system/bin/failsafe/",
26                  "/data/local/"
27          };
28
29          for (final String where : places) {
30              if (new File(where + binaryName).exists()) {
31                  result = true;
32                  android.os.Process.killProcess(android.os.Process.myPid());
33              }
34          }
35
36          return result;
37      }
```

Listing 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

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`

```
9       public static boolean checkInstall(final Context context) {
10          boolean result = false;
11          String[] luckypatcher = new String[]{
12                  // Lucky patcher
13                  "com.dimonvideo.luckypatcher",
14                  // Another lucky patcher
15                  "com.chelpus.lackypatch",
16                  // Black Mart alpha
17                  "com.blackmartalpha",
18                  // Black Mart
```

```
19                  "org.blackmart.market",
20                  // Lucky patcher 5.6.8
21                  "com.android.vending.billing.InAppBillingService.LUCK",
22                  // Freedom
23                  "cc.madkite.freedom",
24                  // All−in−one Downloader
25                  "com.allinone.free",
26                  // Get Apk Market
27                  "com.repodroid.app",
28                  // CreeHack
29                  "org.creeplays.hack",
30                  // Game Hacker
31                  "com.baseappfull.fwd"
32          };
33
34          for (String string : luckypatcher) {
35              if(checkInstallerName(context, string)){
36                  result = true;
37              }
38
39              if (result) {
40                  android.os.Process.killProcess(android.os.Process.myPid());
41              }
42          }
43
44          return result;
45      }
46
47      private static boolean checkInstallerName(Context context, String string) {
48          PackageInfo info;
49          boolean result = false;
50
51          try {
52              info = context.getPackageManager().getPackageInfo(string, 0);
53
54              if (info != null) {
55                  android.os.Process.killProcess(android.os.Process.myPid());
56                  result = true;
57              }
58
59          } catch (final PackageManager.NameNotFoundException ignored) {
60          }
61
62          if (result) {
63              android.os.Process.killProcess(android.os.Process.myPid());
64          }
65          return result;
66      }
```

```
67 }
```

Listing 4.3: Partial Listing

4.1.4 Sideload Detection

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

4.1.5 Signature Check

once in code
save to use signature in code?
`http://forum.xda-developers.com/showthread.php?t=2279813&page=5`

```
15 public class Sideload {
16     private static final String PLAYSTORE_ID = "com.android.vending";
17     private static final String AMAZON_ID = "com.amazon.venezia";
18     private static final String SAMSUNG_ID = "com.sec.android.app.samsungapps";
19
20     public static boolean verifyInstaller(final Context context) {
21         boolean result = false;
22         final String installer = context.getPackageManager().getInstallerPackageName(context.
               getPackageName());
23
24         if (installer != null) {
25             if (installer.startsWith(PLAYSTORE_ID)) {
26                 result = true;
27             }
28             if (installer.startsWith(AMAZON_ID)) {
29                 result = true;
30             }
31             if (installer.startsWith(SAMSUNG_ID)) {
32                 result = true;
33             }
34         }
35         if(!result){
36             android.os.Process.killProcess(android.os.Process.myPid());
37         }
38
39         return result;
40     }
```

Listing 4.4: Partial Listing

4.1.6 Signature

`http://developer.android.com/tools/publishing/app-signing.html`

CONTRA

Remote Verification and Code nachladen

local check whether signature is allowed

```
51      public static boolean checkAppSignature(final Context context) {
52          //Signature used to sign the application
53          static final String mySignature = "...";
54          boolean result = false;
55
56          try {
57              final PackageInfo packageInfo = context.getPackageManager().getPackageInfo(context.
                    getPackageName(), PackageManager.GET_SIGNATURES);
58
59              for (final Signature signature : packageInfo.signatures) {
60                  final String currentSignature = signature.toCharsString();
61                  if (mySignature.equals(currentSignature)) {
62                      result = true;
63                  }
64              }
65          } catch (final Exception e) {
66              android.os.Process.killProcess(android.os.Process.myPid());
67          }
68
69          if (!result) {
70              android.os.Process.killProcess(android.os.Process.myPid());
71          }
72
73          return result;
74      }
```

Listing 4.5: Partial Listing

Remote Signature Verification and Remote COde Loading

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

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

4.2.1 Modify the Library

google

4.2.2 Checken ob ganzer code abläuft und dann nacheinander elemente aktivieren

master1 - testen

damit die ganzen blöcke durchlaufen werden müssen

4.2.3 dynamische Codegeneration

4.3 Prevent Reengineering

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

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

Filesystem

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

Inject bad OPcode or Junkbytes

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

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

Increase headersize

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?

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

Obfuscators/Optimizors definition
remove dead/debug code
potentially encrypt/obfuscate/hide via reflection
https://youtu.be/6vFcEJ2jgOw?t=243

Relfection

https://www.youtube.com/watch?v=Rv8DfXNYnOI
irgendwo erklären

Proguard

https://youtu.be/6vFcEJ2jgOw?t=419
https://net.cs.uni-bonn.de/fileadmin/user_upload/plohmann/2012-Schulz-Code_Protection_in_Android.pdf
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

Dexguard

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

Allatori

http://www.allatori.com/clients/index.php
https://net.cs.uni-bonn.de/fileadmin/user_upload/plohmann/2012-Schulz-Code_Protection_in_Android.pdf
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/6vFcEJ2jgOw

4.3.3 Protectors and Packers

from malware

APKprotect

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

4.3.4 Packers

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=6vFcEJ2jgOw
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

hosedex2jar

https://youtu.be/6vFcEJ2jgOw?t=1776
PoC packer
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 dexClass-

Loader
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

`https://youtu.be/6vFcEJ2jgOw?t=1982`
anti-debug
anti-tamper
appears to be defunct product
little usage/samples ITW
FUNCTION
`https://youtu.be/6vFcEJ2jgOw?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

4.3.5 BANGCLE

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

4.4.1 Service-managed Accounts

`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

art hat masschinen coed
wenn reengineerbar dann nicht gut

4.4.3 Secure Elements

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

5 Evaluation

Evaluation der vorgeschlagenen punkte mit pro cons und umsetzbarkeit

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

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 Prevent Reengineering

5.2.1 Basic Breaks for Common Tools

Filesystem

Inject bad OPcode

Throw exceptions which are different in dalvik than in java

Increase headersize

Endian Tag?

5.2.2 Optimizors and Obfuscators

Relfection

Proguard

Dexguard

Allatori

Dexprotector

5.2.3 Protectors

APKprotect

5.2.4 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-malwa

`html`

hosedex2jar

Pangxie

5.2.5 BANGCLE

5.3 External Improvements

sis is text

5.3.1 Service-managed Accounts

5.3.2 ART

art hat oat files aber die haben dex files

5.3.3 Secure Elements

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

6 Conclusion

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

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

sis is text http://www.digipom.com/how-the-android-license-verification-library-is-lulling-yo
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

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